UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA DE ESTATÍSTICA
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Christian Aurich Zanettini Martins

Uso de processamento de linguagem natural para detecção de alérgenos em alimentos a partir da lista de ingredientes

Florianópolis
2023

Christian Aurich Zanettini Martins

Uso de processamento de linguagem natural para detecção de alérgenos em alimentos a partir da lista de ingredientes

Trabalho de Conclusão de Curso do Curso de Graduação em Sistemas de Informação do Departamento de Informática de Estatística da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Sistemas de Informação.
Orientador: Prof. Elder Rizzon Santos, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Martins, Christian Aurich Zanettini
Uso de processamento de linguagem natural para detecção de alérgenos em alimentos a partir da lista de ingredientes / Christian Aurich Zanettini Martins ; orientador, Elder Rizzon Santos, 2023.
107 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistemas de Informação, Florianópolis, 2023.

Inclui referências.

1. Sistemas de Informação. 2. Inteligência Artificial. 3. Alergia. 4. Processamento de Linguagem Natural. 5. Ontologias. I. Santos, Elder Rizzon. II. Universidade Federal de Santa Catarina. Graduação em Sistemas de Informação. III. Título.

Christian Aurich Zanettini Martins

Uso de processamento de linguagem natural para detecção de alérgenos em alimentos a partir da lista de ingredientes

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Sistemas de Informação" e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação.

Florianópolis, 6 de dezembro de 2023.

Prof. Álvaro Junio Pereira Franco, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Elder Rizzon Santos, Dr.
Orientador

Prof. Roberto Willrich, Dr.
Avaliador
Universidade Federal de Santa Catarina

Rodrigo Rodrigues Pires de Mello, Me.
Avaliador
Universidade Federal de Santa Catarina

Este trabalho é dedicado ao meu avô Antônio Ozi.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais e avós pelas oportunidades e educação concedidas. A todos meus amigos que acompanharam minha trajetória e estiveram presentes mesmo nos momentos mais complicados. Ao meu orientador professor Elder por ter apoiado minhas ideias desde o início, e aos demais professores da graduação que também me motivaram com aulas incríveis. Também gostaria de agradecer aos meus queridos colegas do Laboratório Bridge, que fizeram parte da maior parte dessa jornada.

"We can only see a short distance ahead,
but we can see plenty there that needs to be done." (Turing, 1950)

RESUMO

A evolução da inteligência artificial tem potencializado a automação de tarefas cotidianas, tornando as máquinas cada vez mais integradas às atividades humanas. Este trabalho propõe a aplicação de processamento de linguagem natural e ontologias para aprimorar a leitura e interpretação de rótulos de alimentos, uma tarefa crucial para o acompanhamento nutricional e prevenção de reações alérgicas. Tendo em vista que a prevalência de alergias alimentares tem crescido significativamente no mundo, torna-se essencial a existência de meios precisos para identificação de suas substâncias causadoras nas listas de ingredientes dos produtos alimentícios. O estudo motivou a elaboração de um modelo que, empregando as técnicas mencionadas, facilita a interpretação automatizada de rótulos, analisando e detectando substâncias alergênicas e seus derivados. Além disso, foram conduzidos experimentos para discernir as técnicas mais eficazes dentro da área de PLN para realizar a tarefa de detecção de alérgenos nas listas de ingredientes. Os resultados desses experimentos contribuem para o aprimoramento do modelo e oferecem panoramas para demais estudos, promovendo conscientização e segurança no consumo alimentar de pessoas alérgicas.

Palavras-chave: Inteligência Artificial. Alergia. Processamento de Linguagem Natural. Ontologias.

ABSTRACT

The evolution of artificial intelligence has enhanced the automation of everyday tasks, making machines increasingly integrated into human activities. This work proposes the application of natural language processing and ontologies to improve the reading and interpretation of food labels, a crucial task for nutritional monitoring and prevention of allergic reactions. Given that the prevalence of food allergies has grown significantly around the world, it is essential to have accurate means of identifying their causative substances in the ingredient lists of food products. The study motivated the development of a model that, using the aforementioned techniques, facilitates the automated interpretation of labels, analyzing and detecting allergenic substances and their derivatives. Additionally, experiments were conducted to discern the most effective techniques within the field of PLN to accomplish the task of detecting allergens in ingredient lists. The results of these experiments contribute to improving the model and offer perspectives for other studies, promoting awareness and safety in food consumption for allergic people.

Keywords: Artificial Intelligence. Allergy. Natural Language Processing. Ontology.

LISTA DE FIGURAS

Figura 1 – Evolução da área de PLN 20
Figura 2 – Arquitetura Transformer 24
Figura 3 – Procedimentos gerais de pré-treino e ajuste fino do BERT 26
Figura 4 – Arquitetura FOODS 30
Figura 5 – Fluxograma do desenvolvimento 35
Figura 6 – Fluxograma da etapa de preparação dos dados 36
Figura 7 – Grafo do relacionamento entre classes e indivíduos da ontologia 42
Figura 8 – Matriz de confusão para Levenshtein 49
Figura 9 – Gráfico de métricas para Levenshein 50
Figura 10 – Matriz de confusão para Jaccard 51
Figura 11 – Gráfico de métricas para Jaccard 52
Figura 12 – Matriz de confusão para spaCy 53
Figura 13 – Gráfico de métricas para spaCy 54
Figura 14 – Matriz de confusão para FastText 55
Figura 15 – Gráfico de métricas para FastText 56
Figura 16 – Matriz de confusão para BERT 57
Figura 17 – Gráfico de métricas para BERT 58

LISTA DE TABELAS

Tabela 1 – Matriz de confusão 28
Tabela 2 – Resumo dos Trabalhos 32
Tabela 3 – Exemplo de preparação de dados para o modelo 38
Tabela 4 – Comparação computacional dos algoritmos 47
Tabela 5 – Comparação das métricas dos algoritmos 58

LISTA DE ABREVIATURAS E SIGLAS

ANVISA	Agência Nacional de Vigilância Sanitária
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BRWAC	Brazilian Web as Corpus
CRISP-DM	Cross-Industry Standard Process for Data Mining
CSV	Comma-separated Values
FN	Falso Negativo
FOODKG	Food Knowledge Graph
FP	Falso Positivo
GPT	Generative Pre-trained Transformer
IA	Inteligência Artificial
JSON	JavaScript Object Notation
LFU	Least Frequently Used
MLM	Masked Language Model
NER	Named Entity Recognition
NSP	Next Sentence Prediction
OCR	Optical Character Recognition
OFFF	Ontology of Fast Food Facts
OWL	Ontology Web Language
PLN	Processamento de Linguagem Natural
RDC	Resolução da Diretoria Colegiada
RDF	Resource Description Framework
RE	Resolução
SPARQL	SPARQL Protocol and RDF Query Language
T5	Text-to-Text Transfer Transformer
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
W3C	World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS 15
1.1.1 Objetivo Geral 15
1.1.2 Objetivos Específicos 15
1.2 MÉTODO DE PESQUISA 15
1.2.1 Revisão bibliográfica 15
1.2.2 Obtenção dos dados 16
1.2.3 Preparação dos dados 16
1.2.4 Experimentações 16
1.2.5 Escolha das técnicas 16
1.2.6 Construção do modelo 16
1.2.7 Avaliação e análise do modelo 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 ALERGIAS ALIMENTARES 17
2.2 ONTOLOGIAS 17
2.2.1 RDF 17
2.2.2 OWL 18
2.2.3 SPARQL 18
2.2.4 Raciocinador 18
2.3 PROCESSAMENTO DE LINGUAGEM NATURAL 19
2.3.1 Técnicas de similaridade entre strings 21
2.3.1.1 Distância de Levenshtein 22
2.3.1.2 Índice de Jaccard 22
2.3.1.3 Similaridade de cosseno 22
2.3.2 Embedding 23
2.3.3 Transformer 24
2.3.4 Modelos de PLN 25
2.3.4.1 spaCy 25
2.3.4.2 FastText 25
2.3.4.3 BERT 26
2.3.4.4 **BERTimbau** 27
2.4 AVALIAÇÃO DE MODELOS 27
2.4.1 Matriz de confusão 27
2.4.2 Acurácia 28
2.4.3 Precisão 28
2.4.4 Recall 28
2.4.5 F1-Score 28

3 TRABALHOS RELACIONADOS 29
3.1 THE ONTOLOGY OF FAST FOOD FACTS: CONCEPTUALIZATION OF NUTRITIONAL FAST FOOD DATA FOR CONSUMERS AND SEMANTIC WEB APPLICATIONS 29
3.2 FOODS: A FOOD-ORIENTED ONTOLOGY-DRIVEN SYSTEM 29
3.3 FOODKG: A SEMANTICS-DRIVEN KNOWLEDGE GRAPH FOR FOOD RECOMMENDATION 30
3.4 OUTROS TRABALHOS 31
3.5 ANÁLISE DOS TRABALHOS RELACIONADOS 31
4 DESENVOLVIMENTO 34
4.1 COLETA DE DADOS 35
4.2 PREPARAÇÃO DOS DADOS 36
4.3 ONTOLOGIA 39
4.3.1 Classes 39
4.3.2 Propriedades de objetos 39
4.3.3 Indivíduos 39
4.3.4 Relações semânticas 41
4.3.5 Anotações e rótulos 41
4.3.6 Equivalência semântica 41
4.3.7 Inferências 42
4.4 MODELO DE INTEGRAÇÃO 43
4.5 AVALIAÇÃO 47
4.5.1 Levenshtein 49
4.5.2 Jaccard 51
4.5.3 spaCy 53
4.5.4 FastText 56
4.5.5 BERT 57
5 CONCLUSÃO 60
APÊNDICE A – ARTIGO 66
APÊNDICE B – CÓDIGO IMPLEMENTADO 80

1 INTRODUÇÃO

A definição de inteligência é bastante subjetiva nos mais diversos campos da ciência. Para Inteligência Artificial (IA), não é diferente. Atualmente, a IA abrange uma enorme variedade de subcampos, do geral, até tarefas específicas. “A IA é relevante para qualquer tarefa intelectual; é verdadeiramente um campo universal” (Russell; Norvig, 2013). Turing (1950) idealizou uma definição de IA ao propor o que posteriormente ficou conhecido como Teste de Turing. Sua premissa supõe que uma máquina seja inteligente caso indistinguível de um humano em uma comunicação em linguagem natural entre eles. Assim, tem-se que para obter sucesso nesse teste empírico, uma máquina deve ser capaz de representar conhecimento, raciocinar, aprender e processar linguagem natural.

Segundo Nardi e Branchman (2003), as formas de representação de conhecimento e raciocínio fornecem descrições de alto nível do mundo e podem ser utilizadas na construção de aplicações inteligentes. Nesse contexto, a inteligência é atribuída à capacidade de encontrar consequências implícitas a partir do conhecimento explicitamente representado. Esse conhecimento explícito enquanto factível, propicia o uso das formas convencionais de IA simbólica. Nessa abordagem, a lógica é evidenciada como meio de raciocínio, e uma das formas notáveis dessa conjuntura, são as ontologias. Originado da Filosofia, o termo "ontologia"refere-se ao estudo do ser. Entretanto na computação é denotado por ser um meio de representar conhecimento, sendo definido por Gruber (1995) como “uma especificação explícita de uma conceituação”.

O Processamento de Linguagem Natural (PLN) é uma área interdisciplinar que converge a computação e a linguística, desempenhando um papel crucial na interação entre sistemas computacionais e linguagens humanas naturais, como o português. De acordo com Luger (2013), a implementação de um programa que compreende linguagem natural requer representação de conhecimento e expectativas do domínio abordado, bem como a capacidade de raciocinar efetivamente sobre eles.

As técnicas de PLN, juntamente com ontologias, têm sido aplicadas para resolver uma variedade de problemas. No contexto deste trabalho, elas são empregadas para abordar um problema significativo na saúde: a detecção de alérgenos em listas de ingredientes de alimentos. Essa identificação de alérgenos é fundamental para a prevenção de reações adversas em pessoas com essa sensibilidade. Os rótulos e listas de ingredientes são fontes ricas de informações sobre os alimentos, e muitas vezes estão repletas de detalhes e conhecimento implícitos, tornando a tarefa de identificação manual de alérgenos exaustiva. Dada as certezas e possibilidade de explicitação acerca da relação entre alergias e alimentos, este estudo visa desenvolver um modelo inteligente dedicado a automatizar e otimizar essa tarefa.

1.1 OBJETIVOS

Nas subseções abaixo estão descritos o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Objetivo Geral

Desenvolver um modelo para detecção de substâncias relacionadas a alergias comuns em alimentos, através da aplicação de técnicas de processamento de linguagem natural no texto de listagem de seus ingredientes.

1.1.2 Objetivos Específicos

- Analisar o estado da arte das técnicas de processamento de linguagem natural.
- Aplicar técnicas de processamento de linguagem natural em um conjunto de dados visando a extração semântica.
- Representar através de ontologias, o conhecimento acerca da relação alérgica entre alimentos e as substâncias que os compõem.
- Propor um modelo que integre a ontologia de alérgenos com a semântica extraída dos dados.
- Avaliar a eficácia do modelo quanto a detecção dos alérgenos.
- Comparar sistematicamente o desempenho das diferentes técnicas de processamento de linguagem natural na realização da tarefa.

1.2 MÉTODO DE PESQUISA

Este trabalho é conduzido por um conjunto estruturado de etapas as quais não seguem estritamente a um procedimento metodológico. Com objetivos definidos, o processo foi parcialmente inspirado por metodologias consolidadas na área como o Cross-Industry Standard Process for Data Mining (CRISP-DM). No entanto, conforme discutido por Martinez-Plumed et al. (2021), a natureza exploratória desta pesquisa exige uma abordagem metodológica mais flexível e adaptável aos objetivos específicos do estudo. A seguir tem-se as etapas que compõem o método.

1.2.1 Revisão bibliográfica

Uma revisão da literatura foi realizada, englobando livros e trabalhos acadêmicos da área de estudo e de domínio do problema. Essa etapa foi crucial para aquisição de um entendimento do domínio facilitando a execução das etapas subsequentes e construção

da fundamentação teórica. Também foi buscado identificar métodos e técnicas relevantes. Esta etapa, além de um levantamento da literatura, foi uma análise crítica que ajudou a refinar o objetivo da pesquisa e a traçar um procedimento metodológico adequado. Também foram procuradas lacunas e oportunidades para inovação no desenvolvimento do trabalho.

1.2.2 Obtenção dos dados

Os dados foram pesquisados e coletados em conjuntos de dados públicos disponíveis na *web*, focando em produtos alimentícios e seus ingredientes. A seleção do conjunto foi baseada na relevância, volume e qualidade para assegurar a robustez na fase de modelagem.

1.2.3 Preparação dos dados

Nesta fase, técnicas de pré-processamento foram aplicadas aos dados coletados para otimizá-los, garantindo que estejam em um formato adequado para alimentar o modelo e favorecer a obtenção de resultados precisos e confiáveis.

1.2.4 Experimentações

O modelo é mais do que uma implementação técnica; é uma representação que deve ser submetida a testes para validar sua eficácia e precisão.

1.2.5 Escolha das técnicas

A partir das experimentações, foi possível escolher as técnicas e parâmetros mais apropriados para serem utilizadas no desenvolvimento do trabalho.

1.2.6 Construção do modelo

O modelo foi construído integrando uma base de conhecimentos sobre alimentos utilizando ontologias, e combinando com as características extraídas durante a preparação dos dados. Este modelo visa a identificação de alérgenos nos textos dos produtos alimentícios e é utilizado em conjunto com uma variedade de algoritmos e modelos distintos com propósito comparativo.

1.2.7 Avaliação e análise do modelo

Foi realizada a descrição do processo de avaliação dos resultados obtidos, incluindo a análise crítica das métricas e resultados, e como eles são comparados não apenas entre as diferentes técnicas utilizadas neste estudo, mas também em relação às práticas estabelecidas no estado da arte. Essa abordagem fundamenta a aplicação do modelo e das técnicas empregadas para demais propósitos relacionados.

2 FUNDAMENTAÇÃO TEÓRICA

As áreas de pesquisa relacionadas ao tema do trabalho foram revisadas a fim de estabelecer uma base conceitual necessária para sua compreensão.

2.1 ALERGIAS ALIMENTARES

São reações não-tóxicas do corpo humano como meio de defesa do sistema imunológico. Diferente das intolerâncias alimentares que são reações não imunomediadas, as alergias são classificadas como imunomediadas, isso significa que o organismo da pessoa alérgica reage anormalmente a presença do alimento e produz anticorpos para combatê-lo. Essas reações podem ocasionar em diversos sintomas que podem chegar a ser graves, como é o caso do choque anafilático que é possivelmente fatal.

Grande parte das alergias alimentares são causadas por proteínas presentes em oito alimentos principais: leite, ovos, peixes, crustáceos, amendoins, nozes, soja e trigo. Também podem ocorrer reações cruzadas, onde a sensibilidade às proteínas em uma substância pode desencadear uma reação a outra substância aparentemente não relacionada (Wilkinson, 1998).

2.2 ONTOLOGIAS

Nas ciências da computação e informação, as ontologias são formas de representar conhecimento sobre um domínio. Essa representação acontece por meio da definição de objetos e conceitos que possuem relações entre si e podem ser representados por primitivas de classes e atributos. De acordo com Gruber (1995), esse formalismo caracteriza um conhecimento modelado do domínio que pode ser chamado de universo do discurso. Ele pode ser codificado utilizando linguagens como Ontology Web Language (OWL).

A partir de um universo do discurso, é possível realizar inferências para descobrir conhecimentos explícitos através de raciocínio. Também pode-se realizar consultas declarativas de informações nas ontologias utilizando a linguagem SPARQL Protocol and RDF Query Language (SPARQL). Tudo isso torna as ontologias uma boa forma de armazenamento de dados e conhecimento para aplicar em sistemas inteligentes.

2.2.1 RDF

RDF é um modelo de dados para descrever a estrutura e o conteúdo da web semântica. Ele permite que dados sejam integrados de várias fontes, mesmo que as estruturas subjacentes sejam diferentes. As informações estruturas por RDF são representadas como triplas: sujeito, predicado e objeto. Essas triplas podem ser visualizadas como grafos, facilitando a representação de relações entre diferentes entidades e conceitos.

2.2.2 OWL

É uma linguagem de definição de ontologias web especificada pela World Wide Web Consortium (W3C). Ela desempenha um papel fundamental na construção da Web Semântica, pois facilita a representação de conhecimentos complexos sobre diversos domínios, entidades e as relações entre elas. O OWL auxilia máquinas a interpretarem e responderem de maneira mais sofisticada e inteligente às solicitações de seus usuários, promovendo uma interface mais eficiente e significativa.

Além da versão original do OWL, existem outras versões como o OWL2, que é uma extensão do OWL com maior expressividade e capacidades aprimoradas. O OWL2 oferece novos recursos e funcionalidades que permitem uma modelagem mais detalhada e precisa de ontologias complexas.

O OWL é construído sobre o Resource Description Framework (RDF). Isso significa que o OWL usa o modelo de dados do RDF para fazer afirmações sobre as entidades e relações em uma ontologia. O RDF fornece a base, enquanto o OWL fornece uma camada adicional de expressividade e semântica.

No âmbito do OWL, existem subcategorias, como o OWL-DL (Descrição Lógica), que apresenta um equilíbrio entre a expressividade da linguagem e a eficiência computacional. O OWL-DL é projetado para oferecer um nível notável de expressividade, ao mesmo tempo em que assegura a completude e a decidibilidade das inferências. Esta característica é crucial, pois garante que todas as perguntas formuladas dentro do escopo da ontologia possam ser respondidas de forma conclusiva. Sua robustez o torna adequado para aplicações que necessitam de um raciocínio lógico rigoroso e preciso, permitindo a modelagem detalhada de domínios complexos.

A linguagem é amplamente aplicada em diversos campos que exigem uma representação de conhecimento profunda e interoperável, e nelas contribui para a organização, compartilhamento e utilização eficiente do conhecimento, facilitando a análise, recuperação e gestão de informações.

2.2.3 SPARQL

É um conjunto de padrões que fornece linguagens e protocolos de consulta que possibilita adicionar, remover e recuperar dados de bases de dados de grafos do tipo RDF. As consultas SPARQL são capazes de encontrar correspondências de triplas RDF, utilizar operações matemáticas em suas projeções, aplicar filtragem, ordenações e agrupamentos nos resultados.

2.2.4 Raciocinador

Ontologias bem estruturadas, onde conceitos, entidades e relações são claramente definidas e inter-relacionadas, viabilizam a realização de inferências lógicas e raciocínio au-

tomatizado. Isso contribui para a descoberta de novos conhecimentos e percepções sobre o domínio, e também a validação e verificação da consistência e precisão das informações representadas nas ontologias. Para isso, existem diversos programas que podem ser integrados a essas ontologias para executar essa tarefa.

2.3 PROCESSAMENTO DE LINGUAGEM NATURAL

É uma área que estuda a capacidade dos computadores compreenderem as linguagens naturais humanas através de software. Ela é multidisciplinar e reúne diversas técnicas computacionais para análise automática de texto e representação das linguagens naturais. A capacidade de realizar essas duas tarefas, satisfaz o quesito compreensão. Aplicações com essa proposta podem apresentar diversos desafios, pois os algoritmos ainda são bastante limitados e não reproduzem o entendimento das linguagens naturais com a mesma facilidade que os seres humanos.

Um dos maiores desafios na área, é interpretar a ambiguidade das palavras nos mais diversos contextos. Como destaca Chowdhary (2020), para um humano, cada palavra em um texto desencadeia diversos conceitos semanticamente relacionados, através de memórias e experiências sensoriais. Tudo isso contribui para a realização de tarefas complexas de processamento de linguagem natural, como desambiguação, vinculação textual e rotulagem de papéis semânticos de maneira extremamente rápida e descomplicada.

Existem trabalhos que propõem modelos de integração de processamento de linguagem natural com bases de representação de conhecimento externas que auxiliam no entendimento e checagem da consistência semântica. Um desses modelos é baseado em ontologias, onde essas tarefas são realizadas através da execução de motores de raciocínio.

A evolução do PLN nas últimas décadas tem sido marcada por inovações significativas que transformam a forma como as máquinas entendem e geram texto.

Como ilustra a Figura 1, os modelos de linguagem neural passaram a emergir por volta do início dos anos 2000, marcando transição dos métodos estatísticos tradicionais para abordagens baseadas em aprendizado profundo.

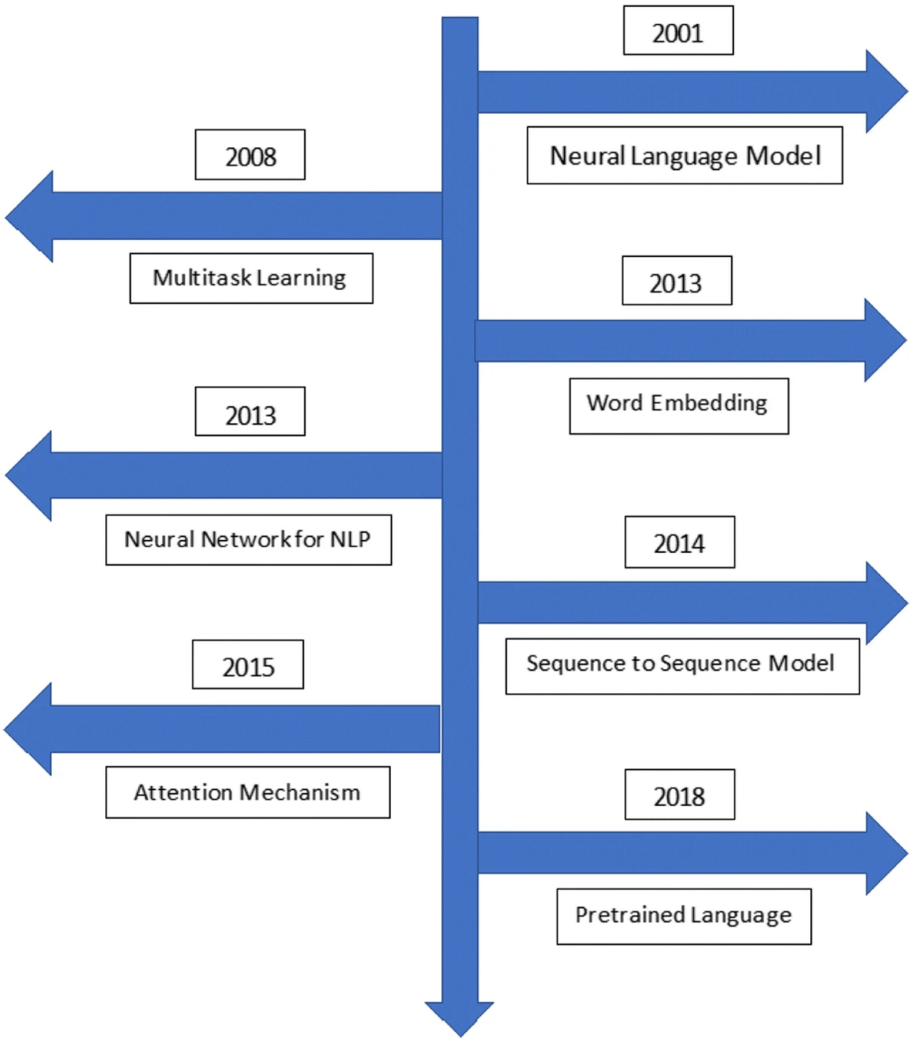

Figura 1 – Evolução da área de PLN

Fonte: Khurana et al. (2022)

A partir de 2008, a comunidade científica passou a explorar a aprendizagem multitarefa, um paradigma que resolve várias tarefas de aprendizado simultaneamente, beneficiando-se assim das informações compartilhadas entre as tarefas relacionadas. Essa abordagem trouxe alguns panoramas para época em relação a modelos linguísticos, como a de que treiná-los em várias tarefas poderia resultar em um melhor desempenho e generalização se comparado a um treinamento em tarefas individuais.

Na década seguinte, em meados de 2013, houve um marco na representação de palavras computacionalmente, a partir da introdução conceito de *word embeddings*. Essa abordagem, que foi aproveitada por modelos como Word2Vec e GloVe, oferece mais riqueza e informação semântica entre as palavras de uma sentença em relação a representações conhecidas até então. No mesmo período a adoção de redes neurais para PLN ganhou impulso, com arquiteturas sendo aplicadas em diversas tarefas e estabelecendo novos padrões de desempenho. Nos dois anos seguintes foram submetidos os conhecidos modelos de sequência a sequência para tarefas como tradução automática. Para isso, foram utilizadas

redes neurais recorrentes, uma inovação que permitiu avanços consideráveis em PLN. A eficácia desses modelos foi ainda mais aprimorada com a introdução dos mecanismos de atenção, conceito que foi fundamental para o desenvolvimento subsequente de modelos atuais e bastante poderosos como os Transformers.

Em 2018, a comunidade testemunhou o surgimento de modelos de linguagem pré-treinados, como BERT e Generative Pre-trained Transformer (GPT). Sua ideia principal é pré-treinar modelos em *corpus* extensos e posteriormente ajustá-los para realização de tarefas específicas. Uma grande variedade de tarefas foram executadas com alto desempenho por modelos que seguem essa abordagem.

Revisitar essa trajetória torna perceptível a série de evoluções incrementais e disruptivas que ocorrem na área de PLN, tornando as máquinas cada vez mais capazes de executar tarefas linguísticas com êxito.

2.3.1 Técnicas de similaridade entre strings

A medição de similaridade entre textos é base das tarefas de processamento de linguagem natural e desempenham um papel importante nos campos de recuperação da informação, respostas automáticas a perguntas, tradução automática, sistemas de diálogo e correspondência de documentos.

O método de medição é descrito por dois aspectos, a distância do texto e a representação do texto, cada uma contando com subcategorias analíticas que enriquecem a compreensão dos textos.

A distância do texto se desdobra em elementos essenciais para uma avaliação minuciosa. Primeiramente, a distância de comprimento funciona como uma ferramenta preliminar, focada na quantificação física do texto, ela realiza realiza uma mensuração baseada no comprimento, enquanto a distância da distribuição amplia a análise, explorando a disposição espacial das palavras possibilita ou outros componentes intratextuais. Já a incorporação da distância semântica procura discernir a proximidade conceitual entre as diversas entidades textuais presentes no texto, fornecendo uma camada adicional interpretativa a ele. Todas essas subcategorias estão essencialmente relacionadas à mesma ideia de distância ou separação dentro do texto (Wang; Dong, 2020).

Por outro lado, a representação do texto é caracterizado como um componente que aborda a compreensão do texto sob perspectivas de representação, análise e entendimento. A versão baseada em *string* serve como um ponto de partida fundamental, tornando a análise na essência literal das sequências de caracteres do texto. Uma abordagem mais refinada é vislumbrar na representação de texto semântico único baseado em *corpus*, onde um *corpus* preexistente de textos é utilizado como referencial para uma exploração mais aprofundada do significado inerente de uma palavra. Mais um passo além, o texto multi semântico traz uma avaliação mais nuançada, suportando a coexistência de múltiplos níveis de interpretação e significado. E por fim, a representação baseada em estrutura

gráfica, que apresenta uma perspectiva visual e estrutural, que ilustra as características e relacionamentos semânticos do texto, possibilitando o entendimento dessas relações e hierarquias textuais.

2.3.1.1 Distância de Levenshtein

A Distância de Levenshtein é uma implementação de uma família de algoritmos conhecidos como distância da edição. Ela representa matematicamente o número mínimo de operações de edição de uma *string*, como inserções, deleções ou substituições necessárias para transformá-la em outra (Castells-Rufas, 2023). Esta técnica é comumente utilizada no processamento de linguagem natural por sua eficiência e simplicidade, operando independentemente de modelos de aprendizado de máquina. A definição do algoritmo é dada pela equação:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{senão.} \end{cases} \tag{1}$$

2.3.1.2 Índice de Jaccard

O índice de Jaccard é uma métrica utilizada para comparar a similaridade entre dois conjuntos. Ele é especialmente útil em casos como comparação de documentos de texto. Como mostram os resultados de Niwattanakul et al. (2013), esse índice se mostra eficaz na medição de similaridade entre palavras, especialmente quando há uma troca de posições de letras nas palavras. No entanto, ela mostra-se ineficaz em detectar palavras com digitação excessiva, onde letras são repetidas.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2}$$

2.3.1.3 Similaridade de cosseno

É uma métrica que modela um texto como um vetor de termos e então a similaridade entre dois textos pode ser calculada pelo cosseno entre os dois vetores. Esses textos podem ser de qualquer tamanhos, sejam frases, parágrafos e até documentos inteiros (Rahutomo; Kitasuka; Aritsugi, 2012).

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3}$$

2.3.2 Embedding

São uma forma de representação de palavras ou frases em espaços vetoriais contínuos de dimensões reduzidas. Essa técnica permite que palavras ou textos sejam convertidos em vetores numéricos, que podem ser usados como entrada para modelos de aprendizado de máquina e aprendizado profundo (Sawicki; Ganzha; Paprzycki, 2023).

Os embeddings podem ser categorizados em contextualizados e não-contextualizados. Essa distinção é essencial no processo de desambiguação de palavras em textos, como pode ser o caso de "Python", que dependendo do contexto pode estar se referindo a uma linguagem de programação de computadores, ou de um gênero de animais répteis.

Pilehvar e Camacho-Collados (2021) apresentam uma comparação crítica entre técnicas convencionais e modelos mais recentes de PLN. Enquanto métodos convencionais como Word2Vec e GloVe tratam palavras como tokens únicos, atribuindo um embedding individual a cada um, modelos mais recentes como Bidirectional Encoder Representations from Transformers (BERT) e GPT adotam uma abordagem baseada em subpalavras, segmentando palavras em unidades menores para gerar embeddings. Diferentes algoritmos de tokenização são empregados para decompor palavras em subpalavras.

A adoção da segmentação em subpalavras em PLN apresenta uma série de vantagens significativas, que são cruciais para a eficiência e eficácia dos modelos linguísticos conforme a seguir descrito.

1. Redução do tamanho do vocabulário: A segmentação em subpalavras contribui substancialmente para a redução do tamanho do vocabulário. Isso acontece ao dividir palavras em unidades menores ou subpalavras, o que simplifica a complexidade do vocabulário e, consequentemente, otimiza o processo de treinamento do modelo.

2. Facilitação no manuseio de palavras novas: Essa abordagem é eficaz no manuseio de palavras que não foram encontradas durante a fase de treinamento. Isso é possível porque qualquer nova palavra pode ser reconstruída e compreendida através de suas subpalavras, para as quais os embeddings já estão disponíveis.

3. Compartilhamento de informações entre palavras semelhantes: A segmentação permite que o modelo compartilhe e generalize informações entre palavras com estruturas ou raízes semelhantes, potencialmente melhorando o entendimento semântico do modelo. Isso é particularmente útil para reconhecer cognatos entre diferentes línguas ou termos que estão lexicalmente relacionados dentro de uma mesma língua.

Essa inovação na maneira de tratar e incorporar palavras permite que modelos de linguagem sejam mais eficientes, flexíveis e capazes de lidar com a diversidade e complexidade das línguas humanas.

De todo modo, os embeddings capturam o contexto semântico das palavras, ou seja, palavras com significados semelhantes são representadas por vetores próximos no

espaço vetorial. Eles ajudam a determinar diferenças e similaridades entre dois textos analisados e são amplamente utilizados em várias tarefas de PLN, como classificação de texto, tradução automática, análise de sentimentos e muitos outros.

2.3.3 Transformer

São uma arquitetura de redes neurais focada principalmente em tarefas de PLN, como tradução automática, modelos de linguagem, classificação de texto. Têm sido base para modelos como BERT, GPT, Text-to-Text Transfer Transformer (T5), entre outros.

Semelhante a outras arquiteturas de redes neurais, a arquitetura original de Transformer, conforme ilustrado na Figura 2, é composta por dois componentes principais: o codificador, localizado à esquerda, que processa a sequência de entrada e a converte em uma representação intermediária; e o decodificador, à direita, que interpreta essa representação para produzir a sequência de saída.

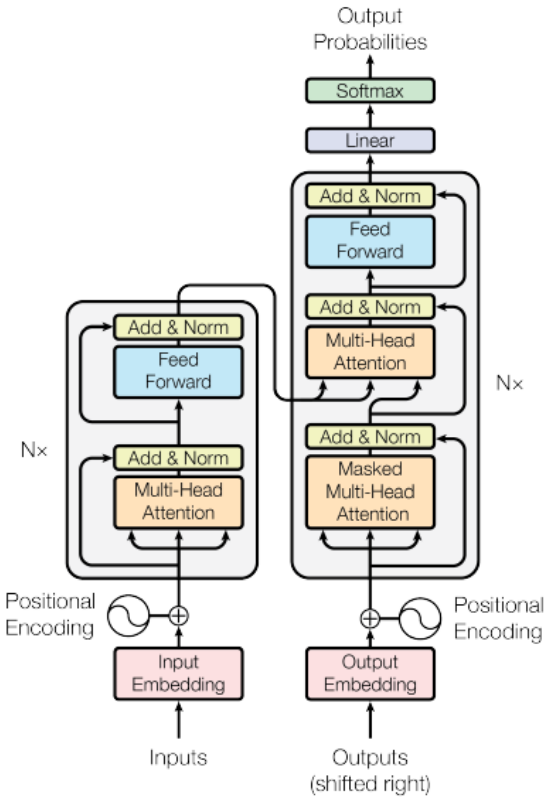

Figura 2 – Arquitetura Transformer

Fonte: Vaswani et al. (2017)

Como apresentado por Vaswani et al. (2017), a arquitetura contém um "mecanismo de atenção". A atenção é descrita como o mapeamento de uma consulta e um conjunto de pares chave-valor para uma saída. Sendo assim, o mecanismo faz com que cada posição de uma sequência de entrada tenha diferentes pesos de atenção para todas as posições na sequência de saída. Na aplicação de PLN, essa distribuição ponderada de atenção permite

que a arquitetura lide mais facilmente com dependências de longo alcance em textos, diferente da arquitetura de redes neurais recorrentes, onde a informação flui sequencialmente e a captura dessas dependências são muito mais desafiadoras.

2.3.4 Modelos de PLN

O PLN envolve a concessão e implementação de modelos, sistemas e algoritmos para resolver problemas práticos de compreensão das línguas humanas. Esses modelos são estruturas computacionais que visam simular, analisar e compreender aspectos variados da linguagem humana sendo essenciais para uma variedade de tarefas complexas, como tradução automática, resposta a perguntas, sumarização, entre outros. Com o advento do aprendizado profundo, os modelos de PLN têm visto avanços significativos, permitindo melhorias notáveis em várias aplicações, como reconhecimento de entidades nomeadas e tradução de idiomas. (Lauriola et al, 2021).

2.3.4.1 spaCy

É uma biblioteca de código aberto para PLN avançado escrita em Python. É projetada para ser de alta performance e fácil de usar, oferecendo as melhores implementações de algoritmos atuais da área. Ela é amplamente utilizada em tarefas como tokenização, lematização, identificação de entidades nomeadas, análise de dependência, entre outras. A spaCy oferece suporte a diversos modelos de *word embeddings*, incluindo vetores de palavras pré-treinados e embeddings contextuais.

2.3.4.2 FastText

É uma biblioteca de aprendizado de máquina de código aberto criada pelo Facebook AI Research Lab, que se especializa em *word embeddings*. Ela trouxe avanços significativos ao campo de PLN ao introduzir métodos que melhoram a representação de palavras e a eficiência e precisão na classificação de texto.

O FastText fornece um modelo Word2Vec pré-treinado para 157 línguas que foi treinado no Common Crawl e na Wikipedia e ao contrário de modelos Word2Vec tradicionais, opera nos níveis de palavras e subpalavras (Young; Rusli, 2019). A técnica utilizada pelo modelo para abordar a tokenização nesses níveis é apresentada por Bojanowski et al. (2017), mostrando que isso pode acontecer a partir da representação de uma palavra como um saco-de-palavras de n-gramas de caracteres. Isso permite que o FastText capture informações morfológicas, tornando-o especialmente poderoso para idiomas com muitas formas de palavras. Graças a isso, ela também pode gerar embeddings para palavras que não aparecem durante o treinamento. Apesar disso, seus *embeddings* são estáticos e não consideram o contexto de toda a sentença. O modelo utiliza uma arquitetura de redes neurais rasas, o que permite treinar modelos rapidamente. Além disso, ele trata cada

documento como uma média de seus *embedding*, permitindo classificações eficientes e a capacidade de lidar com grandes conjuntos de dados e números de classes.

2.3.4.3 BERT

Como introduzido por Devlin et al. (2018), é um modelo de representação de linguagem que foi projetado para pré-treinar representações bidirecionais profundas de texto não rotulado. Como demonstrado à esquerda da Figura 3, o pré-treinamento é realizado com duas tarefas não supervisionadas: Masked Language Model (MLM) e Next Sentence Prediction (NSP). No MLM, alguns tokens de entrada são mascarados aleatoriamente, e o modelo é treinado para predizer esses tokens mascarados com base no contexto bidirecional. Já com NSP, o modelo é treinado para prever se duas sentenças são consecutivas ou não e assim poder entender melhor as relações entre as sentenças.

Seus *embeddings* são contextuais, o que significa que a representação de uma palavra pode mudar com base no contexto em que parece, permitindo que o modelo capture polissemia e nuances textuais.

O BERT reduz a necessidade de arquiteturas específicas para tarefas que exigem pré-treinamento pesado. À direita da Figura 3 é representado o processo conhecido como ajuste fino, onde com apenas uma camada de saída adicional é possível criar modelos para diversas tarefas sem muitas alterações.

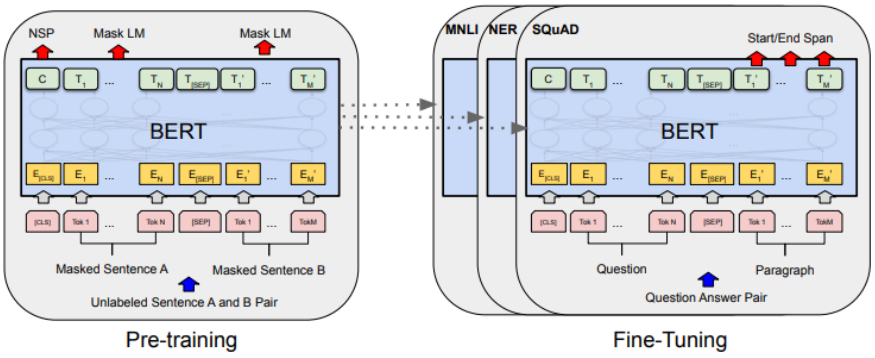

Figura 3 – Procedimentos gerais de pré-treino e ajuste fino do BERT

Fonte: Devlin et al. (2018)

Ele é baseado na arquitetura Transformer e é pré-treinado em um *corpus* composto pelo BookCorpus e pela Wikipedia em inglês, o que totaliza cerca de 3,3 bilhões de palavras. Essa arquitetura é bastante poderosa e auxilia na captura de dependências de longo alcance entre termos. Há duas configurações disponíveis, o BERT Base que possui 12 camadas e 110 milhões de parâmetros, e o BERT Large, com 24 camadas e o total de 340 milhões de parâmetros.

Algumas tarefas específicas que passam por ajuste fino que são amplamente utilizadas são:

- Classificação de Sentença Única;
- Classificação de Pares de Sentenças;
- Resposta a Perguntas;
- Tarefas de Etiquetagem de Sentença Única.

2.3.4.4 **BERTimbau**

Modelo BERT pré-treinado especificamente para o português brasileiro seguindo a arquitetura original. Para os dados de pré-treinamento, foram utilizadas páginas da Wikipédia em português e o *corpus* Brazilian Web as Corpus (BRWAC), que é um conjunto de dados extenso e diversificado para o idioma. O vocabulário gerado no modelo é composto por trinta mil subunidades de palavras e é sensível a maiúsculas e minúsculas. Dois tamanhos de modelo foram treinados, o Base que conta com 110 milhões de parâmetros e 12 camadas, e o Large que utiliza 335 milhões de parâmetros e 24 camadas.

Os modelos BERTimbau foram finamente ajustados para a realização de tarefas como Similaridade Textual de Sentenças e Reconhecimento de Entidades Nomeadas, que de acordo com Souza (2020) apesar de perda de desempenho computacional, os modelos BERTimbau superaram a eficácia do BERT Multilíngue e outros modelo anteriormente publicados.

2.4 AVALIAÇÃO DE MODELOS

É uma etapa crucial que determina a eficácia e aplicabilidade de um modelo em cenários do mundo real, especialmente em tarefas de classificação (Sokolova; Lapalme, 2009).

2.4.1 Matriz de confusão

De acordo com Heydarian, Doyle e Samavi (2022), a matriz de confusão é um método poderoso para analisar um classificador multi-classes em duas dimensões. Ela se responsabiliza por mostrar a distribuição dos acertos e erros das predições em uma única visualização.

Como Ruuska et al. (2018) retrata, em uma matriz de confusão binária, as instâncias corretamente classificadas como pertencentes à classe positiva são denominadas Verdadeiro Positivo (VP), enquanto aquelas corretamente identificadas como pertencentes à classe negativa são chamadas de Verdadeiro Negativo (VN). Em contraposição, as instâncias que, apesar de pertencerem à classe positiva, são erroneamente classificadas como negativas, são conhecidas como Falso Negativo (FN). Da mesma forma, as instâncias da classe negativa que são incorretamente classificadas como positivas são referidas como Falso Positivo (FP).

As categorias VP, VN, FN e FP são fundamentais para a avaliação de desempenho de um classificador. A partir delas, é possível calcular diversas métricas que oferecem uma visão abrangente sobre a eficácia do classificador.

A Tabela 1 demonstra a estrutura de uma Matriz de confusão.

Tabela 1 – Matriz de confusão

	Predição positiva	**Predição negativa**
Real positivo	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Real negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

2.4.2 Acurácia

A acurácia é definida como a razão entre o número de previsões corretas de um modelo e o número total de amostras de entrada. Sendo assim, é uma indicação de quão frequentemente o modelo está correto em suas previsões. É calculada como:

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{4}$$

2.4.3 Precisão

A precisão é a proporção de positivos previstos que são realmente positivos. Isso significa que, de todas as amostras que o modelo previu como positivas, quantas realmente eram positivas. O valor é obtido por:

$$Precisão = \frac{VP}{VP + FP} \tag{5}$$

2.4.4 Recall

O recall, também conhecido como sensibilidade, foca na quantidade de amostras positivas que foram corretamente identificadas pelo modelo. É a proporção de positivos reais que são previstos como positivos, sendo calculada através da fórmula:

$$Recall = \frac{VP}{VP + FN} \tag{6}$$

2.4.5 F1-Score

É a média harmônica entre precisão e recall, por isso é uma métrica que leva em consideração tanto falsos positivos quanto falsos negativos. Definido por:

$$F1 = 2 \cdot \frac{Precisão \cdot Recall}{Precisão + Recall} \tag{7}$$

3 TRABALHOS RELACIONADOS

Durante a revisão bibliográfica foram identificados estudos relevantes que se alinham com a temática e as metodologias empregadas neste trabalho, utilizando procedimentos e ferramentas análogas para alcançar objetivos convergentes.

Para estabelecer uma conexão clara entre este estudo e trabalhos anteriores, foram adotados alguns critérios de avaliação. Entre eles, destacam-se a criação de ontologias e a implementação de processamento de linguagem natural no contexto da alimentação e nutrição. Este capítulo apresenta uma síntese desses estudos, expondo suas abordagens, métodos e contribuições para o campo, fornecendo assim um panorama informativo e uma contextualização do presente trabalho.

3.1 THE ONTOLOGY OF FAST FOOD FACTS: CONCEPTUALIZATION OF NUTRITIONAL FAST FOOD DATA FOR CONSUMERS AND SEMANTIC WEB APPLICATIONS

No contexto contemporâneo, onde o consumo de fast food se tornou uma prática rotineira, a necessidade de informações nutricionais claras e acessíveis é proeminente. Neste cenário, Amith et al. (2021) apresentam uma solução através do desenvolvimento de uma ontologia específica para *fast foods* denominada Ontology of Fast Food Facts (OFFF). Ela foi construída utilizando o software Protégé e a linguagem OWL2. Composta por 413 classes e diversas propriedades, reflete a complexidade e a riqueza de informações necessárias para abranger o universo nutricional dos *fast foods*. A disponibilização de informações nutricionais padronizadas pode influenciar positivamente as escolhas alimentares dos consumidores. Relacionado a isto, um aspecto notável da OFFF é sua atenção a agentes alergênicos que podem compor os ingredientes dos itens de fast food. Ao incorporar informações sobre esses componentes, a ontologia se torna uma aliada valiosa para indivíduos com restrições alimentares ou alergias, oferecendo-lhes auxílio em escolhas alimentares mais saudáveis e seguras.

3.2 FOODS: A FOOD-ORIENTED ONTOLOGY-DRIVEN SYSTEM

Snae e Bruckner (2008) apresentam a proposta de construção de uma ontologia destinada a representar conhecimentos nutricionais e informações tradicionalmente encontradas em rótulos nutricionais, oferecendo uma representação mais rica e acessível das relações entre ingredientes, nutrientes e outras substâncias. O objetivo é auxiliar os usuários na escolha consciente de alimentos a serem consumidos. A ontologia, escrita em e criada com o auxílio do Protégé, é integrada a um sistema denominado FOODS, que considera diversos fatores como sazonalidade, preço e saúde ao fornecer as orientações sobre escolhas alimentares.

A ontologia é capaz de raciocinar e responder perguntas básicas sobre alimentos,

como suas propriedades e doenças relacionadas, e quando combinada com o FOODS, pode responder questões mais complexas, considerando gostos pessoais e necessidades dos usuários.

A pesquisa se destaca no campo de estudo já que grande parte das ontologias têm sido desenvolvidas principalmente para grupos específicos de alimentos ou para extrair informações de receitas.

É destacado que os rótulos nutricionais dos alimentos são regulamentados por lei em diversos países, e a quantidade de pessoas que sabem interpretá-los vem aumentando, por conta do acesso à informação e até ensino básico de nutrição. Porém, os rótulos não são as únicas formas de representar informações nutricionais de alimentos.

A Figura 4 ilustra a arquitetura do sistema.

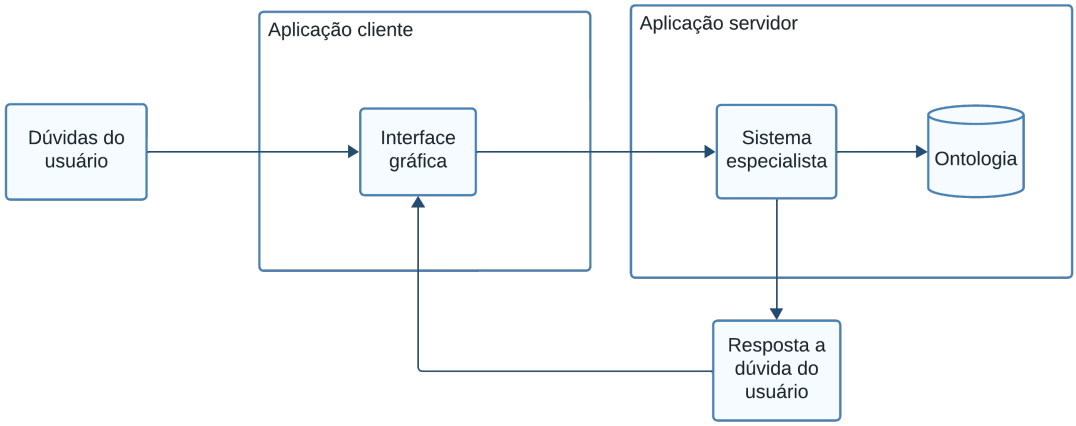

Figura 4 – Arquitetura FOODS

Fonte: Autor do trabalho

3.3 FOODKG: A SEMANTICS-DRIVEN KNOWLEDGE GRAPH FOR FOOD RECOMMENDATION

Haussmann et al. (2019) propõem a construção e aplicação de um Food Knowledge Graph (FOODKG). Ele é um recurso que integra diversas informações relacionadas a alimentos, como receitas, informações nutricionais e ontologias, com o objetivo de recomendar alimentos e receitas saudáveis de maneira personalizada.

Seu processo de construção envolve a coleta de dados de várias fontes como receitas online e um conjunto de dados de nutrição do Departamento de Agricultura dos Estados Unidos. Diversos desafios são enfrentados durante o processo, como dados inválidos, incompletos e ambiguidades. O grafo é construído e estruturado utilizando ontologias, o que permite a explicitação de relações semânticas significativas entre diferentes conceitos,

como receitas e seus ingredientes.

Consultas complexas contam com utilização do SPARQL, facilitando a busca por recomendações precisas que atendem necessidades e preferências alimentares, considerando aspectos como ingredientes disponíveis e alergias alimentares do usuário. Além disso, é explorada a aplicação de agentes cognitivos e técnicas de PLN para permitir que os usuários interajam intuitivamente com o sistema, fazendo perguntas em linguagem natural.

3.4 OUTROS TRABALHOS

Nesta seção, são explorados estudos adicionais que demonstram certa relação com este trabalho.

Vaz (2018) atua na detecção de proteínas e açúcares em alimentos através da análise de informações nutricionais presentes nos rótulos de alimentos. Esta pesquisa é particularmente benéfica para a gestão dietética de consumidores com diabetes ou para aqueles que necessitam de um controle rigoroso da ingestão de proteínas.

Também no intuito de aliar a computação com alimentação e saúde, Clunis (2018) contribui ao desenvolver uma ontologia que cataloga receitas, nutrientes alimentares, interações medicamentosas e condições de saúde. A ontologia possui potencial vasto para auxiliar pacientes no acompanhamento de doenças crônicas, incluindo hipertensão arterial e diabetes, fornecendo uma base de conhecimento para a tomada de decisões dietéticas.

Por fim, Lacson e Long (2006) abordam a identificação automática de alimentos, quantidades consumidas e notas temporais em registros alimentares utilizando PLN. Essa tecnologia promove uma avaliação nutricional em tempo real, simplificando o processo de monitoramento da ingestão alimentar e potencializando a precisão das recomendações nutricionais.

3.5 ANÁLISE DOS TRABALHOS RELACIONADOS

Esta seção demonstra através da Tabela 2 uma breve comparação entre os trabalhos relacionados.

Tabela 2 – Resumo dos Trabalhos

Título do trabalho	**Está no domínio alimentício**	**Utiliza ontologias**	**Utiliza PLN**	**Métricas de avaliação**	**Resultados**
Designing an Ontology for Managing the Diets of Hypertensive Individuals	Sim	Sim	Não	Não possui	Ontologias são eficazes para o gerenciamento de doenças crônicas.
Aplicação de Deep Learning para o reconhecimento de tabelas nutricionais e de ingredientes nos produtos alimentícios	Sim	Não	Sim	Não possui	É possível realizar a classificação de produtos alimentícios pelo seu grau de proteína a partir de imagens de seu rótulo.
The ontology of fast food facts: conceptualization and semantic web applications	Sim	Sim	Não	Veracidade baseada na revisão por especialistas	76,1% dos 103 de declarações da ontologia estavam erradas.
FOODS: A Food-Oriented Ontology-Driven System	Sim	Sim	Não	Não possui	A ontologia e o sistema foram desenvolvidos com sucesso, porém não há avaliação da eficácia.
FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation	Sim	Sim	Sim	F1-Score	95,5%.

Continua na próxima página

Tabela 2 – *Continuação da página anterior*

Título do trabalho	**Está no domínio alimentício**	**Utiliza ontologias**	**Utiliza PLN**	**Métricas de avaliação**	**Resultados**
Natural Language Processing of Spoken Diet Records (SDRs)	Sim	Não	Sim	Acurácia	Identificação de alimentos: 95%; Quantificação de alimentos: 98%; Classificação dos alimentos: 92%; Anotação temporal: 100%.

Ao decorrer do levantamento dos trabalhos relacionados, observou-se que, embora nenhum deles compartilhasse integralmente os objetivos específicos do presente trabalho, eles fornecem uma base teórica e metodológica para a sua realização. As abordagens e técnicas discutidas nos estudos examinados contribuíram para a construção conceitual necessária para o desenvolvimento do trabalho. Esta revisão crítica da literatura existente também destacou lacunas e oportunidades de pesquisa que este trabalho visa abordar. Assim, as fundações estabelecidas pelos trabalhos relacionados são essenciais para a compreensão das estratégias e métodos aplicados nos capítulos subsequentes.

4 DESENVOLVIMENTO

Este capítulo visa delinear o processo de experimentação e os métodos empregados na tarefa de identificar alérgenos em listas de ingredientes de produtos alimentícios.

A complexidade inerente à essa tarefa, dada a diversidade e ambiguidade na forma com que os alérgenos podem ser apresentados nos rótulos, motivou a busca por uma abordagem computacional inteligente para realizá-la. Essa abordagem a ser apresentada é implementada de maneira simbólica. Neste contexto, as ontologias desempenham um papel crucial, funcionando como mecanismo de raciocínio do sistema. E além de poder indicar quais são os alérgenos conhecidos, também auxilia ativamente na desambiguação dos termos. Esse processo é vital para garantir que o sistema identifique corretamente os alérgenos, mesmo quando estes são listados de maneiras diferentes nos rótulos dos produtos, proporcionando assim uma camada adicional de precisão e confiabilidade à detecção de alérgenos realizada.

Para a tarefa de detecção de alérgenos é crucial entender a estrutura das listas de ingredientes e considerar seus contextos. No entanto, é importante salientar que o contexto encontrado em listas de ingredientes é intrinsecamente diferente do encontrado em textos narrativos, diálogos ou conversas. Nas listas de ingredientes o contexto é geralmente mais simples e estruturado. Ele pode incluir hierarquização de ingredientes, onde ingredientes principais podem ter sub-ingredientes listados entre parênteses, ou a inclusão de modificadores que especificam o estado ou a forma de um ingrediente. Esse tipo de contexto é mais direto e menos propenso a ambiguidades.

Foram realizados experimentos e desenvolvimento de um sistema inteligente em Python que aplica técnicas de processamento de linguagem natural para a realização da tarefa de detecção de alérgenos.

O sistema desenvolvido pode ser classificado pela visão de Russell e Norvig (2013) como uma IA que age racionalmente. Isso o caracteriza como um agente racional. Sendo assim, ele atua de maneira a tomar as melhores decisões e alcançar os melhores resultados possíveis na tarefa designada, dadas as informações disponíveis no ambiente.

A proposta de solução a ser desenvolvida segue o procedimento metodológico descrito no trabalho. Uma visão geral dela pode ser vista na Figura 5.

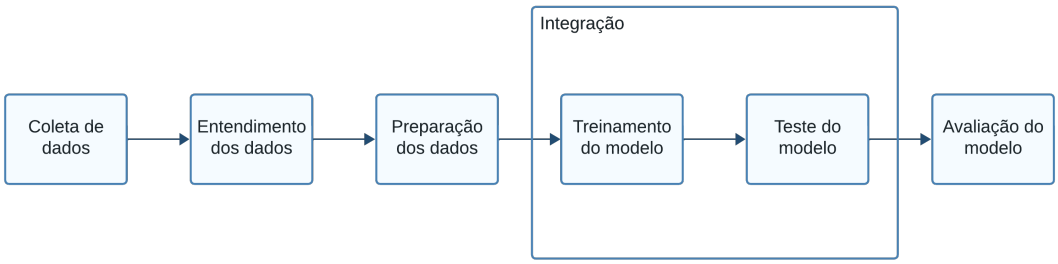

Figura 5 – Fluxograma do desenvolvimento

Fonte: Autor do trabalho

4.1 COLETA DE DADOS

O conjunto de dados empregado no projeto foi obtido a partir do site oficial do Open Food Facts, um banco de dados público e aberto de produtos alimentícios. Esse repositório é notório por seu volume significativo de dados, que são contribuídos por uma comunidade ativa de usuários, pesquisadores e profissionais das áreas relacionadas em todo o mundo. No momento de escrita deste trabalho, a plataforma conta com aproximadamente 3 milhões de produtos registrados.

Para as necessidades específicas deste projeto, foi conveniente buscar apenas os dados que estivessem em conformidade com o contexto linguístico e geográfico de interesse. Esse processo foi complementado com uma filtragem diretamente através da Application Programming Interface (API) fornecida pelo Open Food Facts, a qual permite aos usuários acessar de maneira pragmática.

O processo de filtragem foi direcionado a obter produtos alimentícios cuja informações estivessem disponíveis no idioma português. O conjunto de dados resultante foi adquirido no formato Comma-separated Values (CSV), proporcionando uma base estruturada com cerca de 15 mil produtos para a condução das experimentações planejadas no âmbito deste trabalho.

Os dados podem ser obtidos em outros formatos, incluindo RDF e JavaScript Object Notation (JSON). Essa variedade oferece uma gama de oportunidades para evolução da implementação atual. Por exemplo, é possível automatizar a integração dos dados com a API da plataforma, facilitando e agilizando o acesso e a manipulação dos dados. Além disso, a construção de uma ontologia baseada no modelo de dados original da plataforma. Outra possibilidade é a integração com um banco de dados não relacional como o MongoDB, oferecendo maior flexibilidade e escalabilidade no gerenciamento dos dados.

4.2 PREPARAÇÃO DOS DADOS

O Open Food Facts sendo um banco de dados aberto, permite que usuários registrados na plataforma insiram dados. Essa inserção pode ocorrer por meio de submissão de texto ou de imagens de rótulos de alimentos. No caso de imagens, são empregados métodos de reconhecimento óptico de caracteres para transcrever automaticamente as informações contidas nos rótulos. Contudo, esse processo está sujeito a falhas, sejam elas humanas ou oriundas do software utilizado.

Dessa forma, é imprescindível a realização de um pré-processamento dos dados antes de sua utilização. Os dados obtidos no formato CSV foram manipulados utilizando a biblioteca pandas.

A Figura 6 apresenta uma visão geral do processo de preparação de dados.

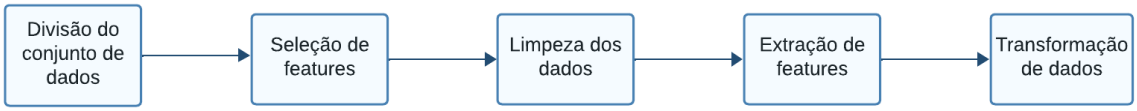

Figura 6 – Fluxograma da etapa de preparação dos dados

Fonte: Autor do trabalho

Inicialmente foi realizada a divisão do conjunto de dados em dois grupos: treinamento e teste. O conjunto de treinamento tem como função primordial de enriquecer a ontologia, populando-a com novos indivíduos derivados dos alérgenos iniciais. Esse conjunto é a base sobre a qual o modelo é construído e refinado.

Já o conjunto de teste detém produtos que não constam no conjunto de treinamento e é utilizado para testar a eficácia do modelo treinado. Para essa divisão foram selecionados 30 produtos alimentícios que foram categorizados manualmente entre fácil, médio e difícil. Considerando a grande quantidade de dados com ruídos presentes no conjunto e da complexidade das inferências necessárias, foram criadas essas categorias que fazem alusão a dificuldade que o sistema pode enfrentar para predizer as saídas corretas. Esse conjunto teve também manualmente criada uma *feature* que representa o gabarito da detecção de alérgenos. O gabarito é essencialmente uma representação precisa dos alérgenos que devem ser identificados nos produtos. Ele serve como referência, permitindo uma avaliação objetiva do desempenho do modelo. O pré-processamento do conjunto de treinamento iniciou-se com a seleção de *features*. Optou-se por manter exclusivamente a coluna da lista de ingredientes, uma vez que componentes do modelo são pré-treinados, e o objetivo não é realizar uma mineração de todo o conjunto, e sim focar na detecção de alérgenos através dessa *feature*. Essa decisão ocasiona em um conjunto de dados mais enxuto e que

favorece a performance do processamento computacional. Subsequentemente foi realizada a limpeza dos dados através das seguintes operações:

- Eliminar instâncias com listas de ingredientes nulas;
- Transformação de todo o texto para letras minúsculas;
- Remoção de espaços em branco nas extremidades das strings;
- Remoção de diacríticos
- Utilização de expressões regulares para remover caracteres especiais e sequências de vírgulas e espaços

A remoção de *stopwords*, embora habitual em projetos de processamento de linguagem natural, não compõe a limpeza de dados deste trabalho. Essa técnica visa eliminar palavras consideradas de pouca relevância para a análise semântica, como artigos, preposições e conjunções. Sua aplicação pode beneficiar a execução da tarefa de detecção de alérgenos, porém exige uma abordagem criteriosa. Algumas palavras frequentemente classificadas como *stopwords* na língua portuguesa, como "de", assumem um papel crucial no auxílio da interpretação dos ingredientes. Por exemplo, o alimento "leite de coco" perde seu significado se o "de" for removido, podendo levar a uma interpretação errônea, onde "leite" é identificado como um alimento separadamente de "coco" caso o restante do pré-processamento não esteja adequado.

No estágio final de preparação de dados, foi procedida a extração de *features*, empregando técnicas de processamento de linguagem natural para isolar termos relevantes nas listas de ingredientes. Para essa tarefa foi elaborada uma expressão regular para segmentar os ingredientes, utilizando vírgulas e a conjunção 'e' como delimitadores.

Durante a fase experimental, foram exploradas diversas outras técnicas de PLN para compor a transformação dos dados e aprimorar a extração de *features*. Embora não tenham sido incorporadas à versão final do modelo, elas apresentam um potencial significativo para seu refinamento.

Primeiramente foram experimentadas técnicas de normalização textual, para reduzir palavras em uma forma base, preservando o significado essencial. Elas auxiliam a lidar com variações morfológicas dos ingredientes, já que fornecem uma padronização das entradas e, por conseguinte, otimizam a correspondência com a lista de alérgenos conhecidos. A lematização, uma dessas técnicas, seria capaz de realizar a associação de "amêndoa" e "amendoado", sendo que ambas explicitam que o alimento contém amêndoas. Essa anotação semântica traz mais flexibilidade nas comparações de identificações.

Bastante semelhante a lematização, também foi experimentada a stemização, um método que corta as extremidades das palavras e apesar de mais rudimentar, comumente é utilizada em cenários com recursos computacionais limitados e dados menos complexos.

Por ser mais simples, também está mais suscetível a falhas dentro do modelo, já que pode por exemplo considerar após a limpeza de texto, que as palavras “amendoim” e “amêndoa” se refiram ao mesmo alimento, o que não seria uma comparação válida.

Além disso, uma parte integral da preparação dos dados foi a preocupação em assegurar a correta separação dos ingredientes por vírgulas. Visando a inserção automática de vírgulas onde faltavam, devido a erros de digitação ou falhas de Optical Character Recognition (OCR), foram experimentadas outras duas técnicas de PLN. A primeira, o Named Entity Recognition (NER), que é o processo de identificar e classificar elementos-chave em textos em categorias predefinidas, como nomes de pessoas, organizações, locais e, no caso deste estudo, ingredientes. E também o POS-Tagging, que envolve a identificação das classes gramaticais. Para essa segunda técnica, foi tentado estabelecer heurísticas para generalizar a identificação de ingredientes a partir das classes gramaticais, porém mostrou-se uma tarefa complexa, já que nem sempre ingredientes são somente um substantivo, também podem assumir outras formas, como serem palavras compostas com preposições. Além disso, a presença de uma palavra que atenda a heurística estabelecida não garante que ela represente um ingrediente, o que limita a precisão dessa abordagem. Isso torna o NER uma ferramenta mais robusta para a identificação precisa de ingredientes. No entanto, ambas as técnicas poderiam contribuir significativamente a estabelecer as delimitações entre ingredientes, facilitando a criação de *features* relevantes para o modelo de detecção de alérgenos.

No contexto do trabalho é relevante compreender como os dados brutos são transformados em dados úteis para a execução da tarefa. A Tabela 3 ilustra exemplos práticos dessa transformação, mostrando como as listas originais de ingredientes são processadas e convertidas em *features* estruturadas como vetores em Python.

Tabela 3 – Exemplo de preparação de dados para o modelo

Lista de ingredientes original	**Feature resultante da preparação de dados**
Tomate, cebola, proteína texturizada de soja, sal, amido modificado, extrata de adura alho, salsa, acidulante ácido lático e aroma idêntico ao natural de bolonhesa.	[’tomate’, ’cebola’, ’proteina texturizada de soja’, ’sal’, ’amido modificado’, ’extrata de adura alho’, ’salsa’, ’acidulante acido latico’, ’aroma identico ao natural de bolonhesa’]
Aveia em flocos, flocos de cevada, açúcar demerara, extrato de malte, fibra de trigo, flocos de arroz, açúcar mascavo, maltodextrina, coco seco, flocos de milho, polpa de banana, cacau em pó, farinha de soja, gergelim, flocos de soja, óleo de girassol, linhaça, mel, aroma idêntico ao natural de chocolate maltado e antioxidante tocoferol (natural).	[’aveia em flocos’, ’flocos de cevada’, ’acucar demerara’, ’extrato de malte’, ’fibra de trigo’, ’flocos de arroz’, ’acucar mascavo’, ’maltodextrina’, ’coco seco’, ’flocos de milho’, ’polpa de banana’, ’cacau em po’, ’farinha de soja’, ’gergelim’, ’flocos de soja’, ’oleo de girassol’, ’linhaca’, ’mel’, ’aroma identico ao natural de chocolate maltado’, ’antioxidante tocoferol natural’]

4.3 ONTOLOGIA

Foi construída uma ontologia para representar o domínio de alérgenos alimentares. Ela é codificada seguindo a especificação RDF e incorporando o vocabulário e construções semânticas oferecidas pela OWL.

A ontologia é composta por classes, propriedades de objetos e indivíduos que representam conceitos relacionados a alimentos e alérgenos alimentares.

4.3.1 Classes

As classes que compõe a ontologia são:

- Alimento: Classe base que representa um superconjunto de todos os alimentos
- Alergeno: Subclasse de Alimento, denotando os alérgenos alimentares
- AlergenoPorDerivacao: Subclasse de Alimento, representando alimentos que são alérgenos devido à sua derivação de outros alérgenos
- Crustaceo: Subclasse de Alergeno, indicando que alimentos de origem animal do subfilo Crustáceo são alérgenos

4.3.2 Propriedades de objetos

eDerivadoDe é uma propriedade que relaciona um alimento a outro do qual é derivado, com domínio e alcance definidos pela classe Alimento.

4.3.3 Indivíduos

Os indivíduos iniciais da ontologia são baseados na resolução Resolução da Diretoria Colegiada (RDC) 26/2015 da Anvisa, que cataloga os principais alimentos causadores de alergias.

1. Trigo, centeio, cevada, aveia e suas estirpes hibridizadas.
2. Crustáceos.
3. Ovos.
4. Peixes.
5. Amendoim.
6. Soja.
7. Leites de todas as espécies de animais mamíferos.

8. Amêndoa (Prunus dulcis, sin.: Prunus amygdalus, Amygdalus communis L.).
9. Avelãs (Corylus spp.).
10. Castanha-de-caju (Anacardium occidentale).
11. Castanha-do-brasil ou castanha-do-pará (Bertholletia excelsa).
12. Macadâmias (Macadamia spp.).
13. Nozes (Juglans spp.).
14. Pecãs (Carya spp.).
15. Pistaches (Pistacia spp.).
16. Pinoli (Pinus spp.).
17. Castanhas (Castanea spp.).
18. Látex natural.

É possível perceber que a lista conta com algumas ambiguidades que precisaram ser abstraídas para simplificação do modelo. É o caso da omissão dos nomes científicos dos alimentos. Já algumas características foram incorporadas como os Crustáceos, que sendo um subfilo do reino animal, engloba diversas espécies que podem aparecer isoladamente nas listas de ingredientes.

Sendo assim, indivíduos resultantes após a análise foram:

- Amendoim
- Amêndoa
- Aveia
- Avelã
- Camarão
- Caranguejo
- Castanha
- Castanha-de-caju
- Castanha-do-brasil
- Castanha-do-pará
- Centeio

- Cevada
- Lagosta
- Lagostim
- Leite
- Látex natural
- Macadâmia
- Nozes
- Ovo
- Pecã
- Peixe
- Pistache
- Siri
- Soja
- Trigo

4.3.4 Relações semânticas

A classe AlergenoPorDerivacao é definida como qualquer instância de Alimento que é derivado de um Alergeno.

4.3.5 Anotações e rótulos

Por convenção, cada classe e indivíduo na ontologia é anotado com um rótulo que facilita consultas e oferece flexibilidades.

4.3.6 Equivalência semântica

O indivíduo Castanha-do-brasil é declarado como equivalente de Castanha-do-pará através do predicado sameAs pois ambos são nomes conhecidos para o mesmo alimento e podem aparecer nos rótulos dos alimentos intercambiavelmente.

A Figura 7 apresenta a visão geral da ontologia.

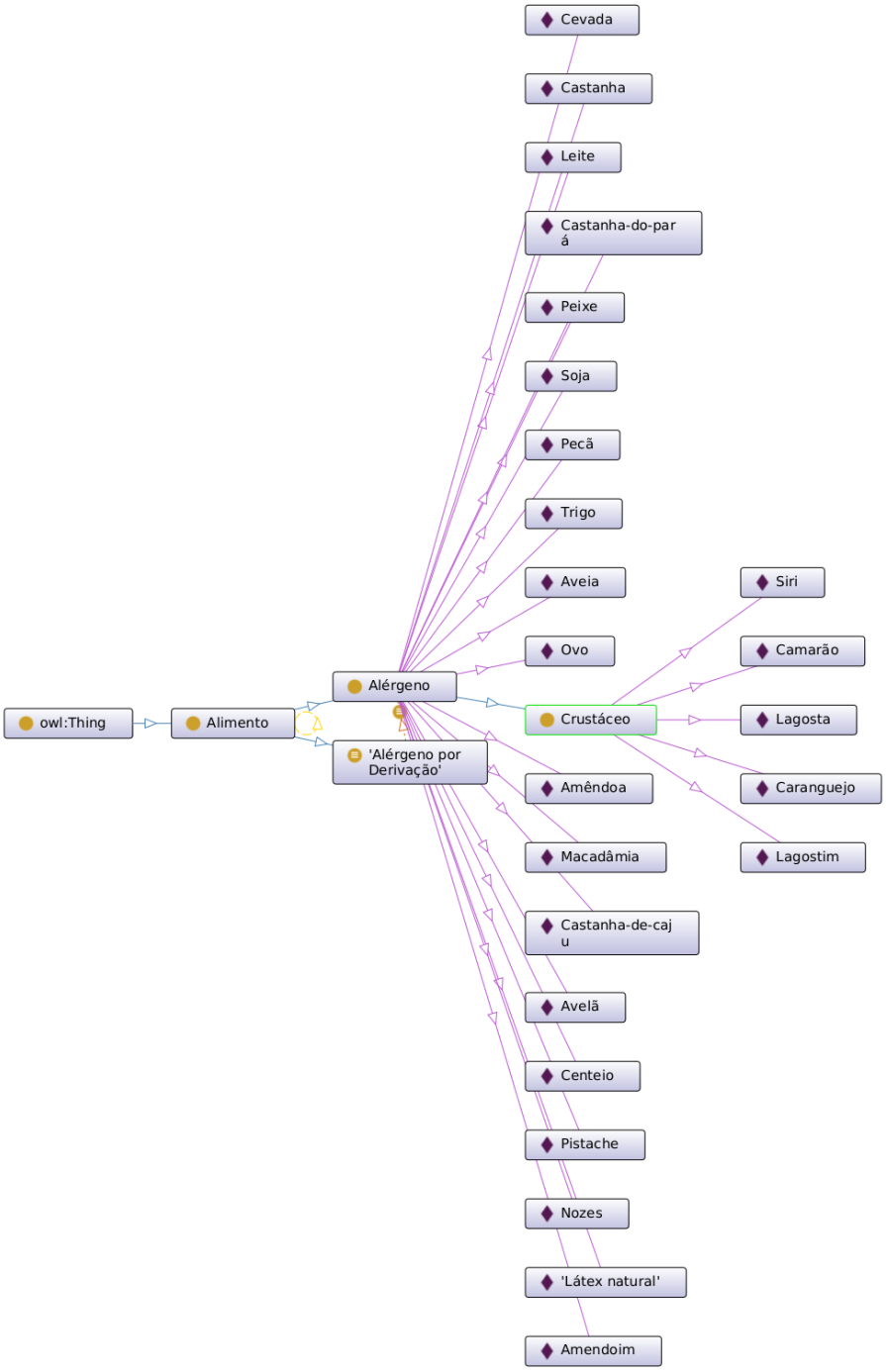

Figura 7 – Grafo do relacionamento entre classes e indivíduos da ontologia

Fonte: Autor do trabalho

4.3.7 Inferências

Todos os indivíduos que possuam a propriedade de objeto descrevendo que o mesmo é um derivado de um indivíduo do tipo Alergeno, é inferido como do tipo AlergenoPorDerivacao.

4.4 MODELO DE INTEGRAÇÃO

Optou-se por realizar uma incorporação prévia de alérgenos na ontologia baseado na heurística de que ingredientes cujas strings que contém nomes de alérgenos já presentes na ontologia são inseridos. Esse procedimento amplia o conhecimento inicial sobre o ambiente e o domínio do problema, proporcionando uma base mais robusta para a identificação e análise de alérgenos.

Contudo, a simplicidade da heurística pode ocasionar em adições equivocadas devido a nuances não capturadas durante a limpeza de dados. Também há casos em que um sistema com uma ontologia previamente populada a partir de uma fonte de dados poderá ter dificuldade de identificar, como é o caso do ingrediente "leite de arroz", pois como é descrito na tabela da Agência Nacional de Vigilância Sanitária (ANVISA), todos os leites alergênicos são os derivados de animais mamíferos, além dos derivados de outros alergênicos como a soja.

Portanto, o leite de arroz pode não ser considerado um alérgeno já que não é de origem animal e nem derivado de algum alérgeno. Outro obstáculo morfológico é o "Queijo" que não é uma locução nominal, então não possui nenhum modificador explícito no substantivo. Sendo assim, sua associação com leite é entendida semanticamente com base em um conhecimento contextual dos falantes da língua.

Mais um caso complexo pode ser o óleo de soja, pois apesar de ser um derivado da soja que é um alimento alergênico, não significa necessariamente que irá desencadear reações alérgicas a uma pessoa, isso é subjetivo, pois como descrito pela Anvisa (2017) na Resolução (RE) 1112, após o refinamento dos óleos, a quantidade de proteína agente causadora da alergia presente é praticamente insignificante.

Porém, como as ontologias nesse trabalho estão sendo abordadas a partir de conhecimentos certos, foram eliminadas variáveis probabilísticas de sua modelagem, esses alimentos estão sendo considerados alérgenos por serem derivados e existir alguma chance de causar alergia mesmo que pouca. Sem todos esses conhecimentos descritos serem definidos por um agente especialista, o uso dessas heurísticas linguísticas resulta em um modelo mais trivial.

A ontologia conforme é populada não está imune a certos erros, principalmente devido à extração de dados ser imperfeita. Esse desafio não é exclusivo deste trabalho, pois conforme analisado por Martins e Dos Reis (2019), até ontologias amplamente difundidas como a DBPedia enfrentam problemas semelhantes. Assim como na DBpedia, onde as informações são automaticamente extraídas da Wikipédia e podem conter imprecisões, a ontologia do presente trabalho também pode refletir as imperfeições do conjunto de dados utilizado.

O raciocinador utilizado no modelo foi o HermiT. Ele foi escolhido por ter conformidade com a OWL 2 Direct Semantics, garantindo que está de acordo com os padrões atuais e bem aceitos da comunidade. Como exposto por Shearer et al. (2008), ele também

conta com diversas otimizações de desempenho e mostra ser capaz de classificar ontologias que outros raciocinadores baseados em lógica descritiva não conseguem, além de ser tão eficiente em termos de desempenho quanto. É o raciocinador padrão da biblioteca owlready2, que cria uma interface entre programas Python e ontologias OWL. O HermiT é compatível com SPARQL, a linguagem de consultas utilizada neste trabalho para integrar a ontologia ao sistema.

Inicialmente, foi cogitado o envio de todos os ingredientes para a ontologia por meio de consultas SPARQL, aproveitando o poder dos *reasoners* atuais para realizar a detecção de alérgenos diretamente na ontologia. No entanto optou-se por realizar a detecção através do programa Python para usufruir das bibliotecas especializadas disponíveis e de outras técnicas utilizadas em PLN para se ter um controle mais granular sobre o processo de comparação e sobre o tratamento de possíveis erros e exceções.

Sendo assim, foi desejado que após o raciocinador fosse ligado, o resultado das inferência fossem consultadas pelo programa. O conhecimento que foi desejado recuperar da ontologia, é a de quais alimentos existentes em seu universo são alérgenos. Tendo isso, habilita o sistema a aplicar técnicas de PLN para detectar os alérgenos nas listas de ingredientes.

A detecção de alérgenos ocorre por meio de uma comparação de *strings*, utilizando três técnicas distintas de cálculo de similaridade para aumentar a precisão na identificação devido aos erros de texto provindos de digitação e OCR. Seria possível realizar o tratamento desses erros na etapa de preparação de dados, apenas necessitaria utilizar o *corpus* de um modelo como do FastText para checar palavras semelhantes em um dicionário, porém essa abordagem foge da proposta metodológica do trabalho.

Os cálculos de similaridade resultam em um valor limiar que varia de 0 a 100. As técnicas empregadas são: Distância de Levenshtein, Índice de Jaccard e Similaridade de Cosseno com *embeddings* gerados pelos modelos spaCy, FastText e BERT. Cada uma dessas técnicas é implementada utilizando bibliotecas para simplificar a codificação, otimizar o desempenho e precisão dos cálculos.

Antes do início da detecção de alérgenos no conjunto de testes, o sistema executa uma fase de otimização de hiperparâmetros. Durante esta fase, os três algoritmos são testados com todos os possíveis limiares de similaridade. O objetivo é identificar qual limiar proporciona o maior valor da métrica F1-Score para cada algoritmo. Considerando a relevância de um alto recall em um modelo voltado ao domínio das alergias alimentares, para assegurar a detecção abrangente de alérgenos, esta métrica inicialmente se apresenta como uma candidata de grande potencial a orientar a otimização do modelo. No entanto, um foco exclusivo no recall pode levar a um desbalanço, fazendo com que o modelo tenda a adotar um limiar de similaridade 0 como parâmetro. Essa sensibilidade excessiva aumentaria a incidência de falsos positivos, uma consequência indesejável na identificação de alérgenos. Para mitigar esse risco, tornou-se necessário avaliar também a precisão. Assim,

a escolha recaiu sobre a otimização com base no F1-Score, uma métrica que harmoniza de maneira eficiente o equilíbrio entre recall e precisão, atendendo às necessidades específicas do modelo.

A Distância de Levenshtein é calculada pela biblioteca thefuzz, que oferece uma implementação eficiente do algoritmo. Sua complexidade isolada é dada por $O(n \cdot m)$ onde n e m são o números de caracteres nas *strings* a serem comparadas.

O Índice de Jaccard é implementado manualmente no código, utilizando a biblioteca nativa de operações de conjuntos em Python, que são eficientes e tem complexidade isolada aproximada de $O(n+m)$ onde n e m representam os tamanhos dos dois conjuntos a serem comparados, da mesma maneira que o algoritmo anterior.

Já a partir da similaridade de cosseno com *embeddings* inicia-se a utilização técnicas mais completas que utilizam modelos de aprendizado de máquina. Essa métrica é calculada pelos três próximos métodos, sendo os dois primeiros não contextuais e o último contextual.

O primeiro desses métodos utiliza o modelo *pt_core_news_md* pré-treinado em português, disponibilizado pela biblioteca spaCy. Este modelo vem com vetores de palavras GloVe incorporados, facilitando a obtenção de representações vetoriais de palavras e sentenças. Ao alimentar o modelo com sentenças, ele retorna *tokens* prontos para comparação e análise subsequente.

Na avaliação de desempenho, a arquitetura de tamanho médio (md) exibiu uma performance superior à versão de tamanho grande (lg), demonstrando uma diferença notável de eficácia. No entanto, é essencial mencionar uma limitação do modelo spaCy em português: seu pré-treinamento é baseado principalmente em dados de notícias[1]. Esta característica sugere que o modelo possa apresentar uma especialização e sensibilidade mais acentuadas para domínios relacionados a notícias, o que pode não estar totalmente alinhado com o domínio específico de alimentos explorado neste estudo. Como consequência, é possível que algumas palavras pertinentes ao contexto alimentar não sejam adequadamente reconhecidas pelo modelo, resultando, em alguns casos, em valores de similaridade igual a zero.

Já como segundo gerador de *embeddings* não contextual, empregou-se o modelo FastText pré-treinado com Word Vectors em português. O processo envolve a decomposição de cada sentença em palavras individuais. Para cada palavra, o modelo FastText é utilizado para obter um vetor de *embedding* correspondente.

A estratégia adotada para representar uma sentença completa foi calcular a média dos *embeddings* de todas as palavras na sentença. Este método resulta em um vetor consolidado que captura a essência semântica da sentença como um todo, facilitando análises e comparações subsequentes no contexto de processamento de linguagem natural.

Por último, para os *embeddings* no modelo BERT, foi utilizada a biblioteca torch.nn.functional, que oferece funções otimizadas para operações com tensores e a

[1] Disponível em: <https://spacy.io/models/pt> Acesso em: 12 de outubro de 2023

biblioteca transformers da Hugging Face, que fornece uma implementação eficiente e fácil de usar do modelo BERT e seu tokenizador.

O modelo BERT em específico que foi utilizado é o BERTimbau, que é um modelo pré-treinado para a língua portuguesa brasileira. A arquitetura escolhida foi a *large* para estender sua capacidade de raciocínio.

Os vetores que serão utilizados para calcular a similaridade de cosseno são representados por *embeddings* BERT que são criados da seguinte maneira:

Primeiro é necessário tokenizar as sentenças utilizando o BERT. As sentenças no contexto desse trabalho são as *strings* dos ingredientes.

A configuração utilizada na tokenização é a seguinte:

Através do parâmetro padding=True é garantido que todas as sequências de tokens tenham o mesmo comprimento, adicionando tokens de preenchimento quando necessário.

truncation=True trunca as sequências de token que excedem o valor definido em max_length que é 128.

Essa tokenização é convertida em tensores compatíveis com a biblioteca PyTorch por conta do parâmetro return_tensors="pt".

Tendo esses tensores, é possível passá-los como parâmetro para o modelo BERT que após percorrer a diversas camadas de transformadores, retorna outro tensor conhecido como *forward output*. Dessa saída extraímos o atributo *last_hidden_state*. Ele contém os *embeddings* dos tokens, que são representações vetoriais densas. Os *embeddings* finais são calculados como a média dos *embeddings* de token no último estado oculto da sequência de tokens de entrada, resultando em um vetor de *embedding* para cada sentença. Dessa maneira, é capturada a essência geral da informação contida em todos tokens da sequência e proporciona uma representação simplificada.

Essa criação de *embeddings* é bastante custosa computacionalmente. Sua complexidade computacional pode ser definida por $O(w \cdot d^2 + d)$, para sequências de comprimento m e dimensão de *embedding* n. Isso ocorre devido ao fato dos transformadores processarem cada token em relação a todos os outros tokens na sequência, resultando em uma complexidade quadrática em relação ao comprimento da sequência.

Devido à vasta quantidade de operações que podem ser executadas dependendo do tamanho do conjunto de testes, foi criado um *cache* do tipo Least Frequently Used (LFU) para armazenar os *embeddings* desses três métodos. Sendo assim, os *embeddings* menos frequentemente utilizados são removidos do *cache* quando não houver mais espaço para inserir novos valores. Além disso, o *cache* é sistematicamente limpo sempre que ocorre a mudança do método em execução, garantindo o devido gerenciamento de memória primária.

A Tabela 4 compara os métodos através de características computacionais:

Tabela 4 – Comparação computacional dos algoritmos

Algoritmo	Complexidade	Necessita de modelo treinado	Tipo de embedding
Levenshtein	$O(n \cdot m)$	Não	-
Jaccard	$O(n + m)$	Não	-
spaCy	$O(d)$	Sim	Não contextual
FastText	$O(w \cdot d + d)$	Sim	Não contextual
BERT	$O(w \cdot d^2 + d)$	Sim	Contextual

'n': tamanho da string do alérgeno;
'm': tamanho da string do ingrediente;
'd': dimensão dos vetores de embedding;
'w': número de palavras de uma sentença.

Para cada algoritmo, a complexidade computacional é influenciada pela operação mais custosa. No contexto dos modelos de linguagem como spaCy, FastText e BERT, a geração de *embeddings* e o cálculo da similaridade de cosseno são as etapas mais exigentes. Para spaCy e FastText, os *embeddings* são comumente recuperados de modelos pré-treinados, o que pode ser considerado uma operação de tempo constante. As complexidades expostas nessa tabela representam a complexidade inerente de cada método de comparação, no entanto, a complexidade total também depende do número de ingredientes e alérgenos a serem comparados, pois o algoritmo será aplicado para cada par de ingredientes e alérgenos.

No caso do *embedding* já constar no *cache*, a complexidade dos três métodos baseados em *embedding* se tornam $O(d)$, já que a recuperação de dados no *cache* pode ser considerada constante, fazendo com que os algoritmos dependam apenas do cálculo da similaridade de cosseno.

Para cada técnica, um limiar de similaridade é definido, e os ingredientes cuja similaridade com um alérgeno conhecido é maior que o limiar são identificados como alérgenos. Este processo é realizado para todos os produtos do conjunto de dados de teste, e os resultados são persistidos e utilizados posteriormente para a avaliação do desempenho do sistema.

Anteriormente foi experimentada uma abordagem que de invés de realizar um povoamento prévio da ontologia com o algoritmo simples de comparação literal, já partisse para as detecções com os algoritmos de PLN apenas com os alérgenos catalogados pela ANVISA. Dessa maneira as inserções de novos alérgenos na ontologia deveria acontecer após uma validação de aprovação por comitê a partir do melhor limiar de cada algoritmo. Porém os resultados obtidos foram muito semelhantes ao modelo atual.

4.5 AVALIAÇÃO

A avaliação do desempenho do sistema desenvolvido utiliza métricas que possibilitam uma análise objetiva quanto sua eficácia na detecção de alérgenos.

Para a avaliação, foram considerados:

- Verdadeiros positivos: Alérgenos que foram identificados corretamente. Ou seja, ingredientes que constam no gabarito de alérgenos e foram corretamente preditos pelo sistema como tal.
- Falsos positivos: Ingredientes que foram incorretamente identificados como alérgenos. São ingredientes que, apesar de serem preditos como alérgenos, não são alérgenos de acordo com o gabarito.
- Falsos negativos: Alérgenos que não foram identificados. São alérgenos que estão presentes na lista de gabarito, mas não foram identificados pelo modelo.
- Verdadeiro negativo: Ingredientes que foram corretamente identificados como não-alérgenos. São ingredientes que não são alérgenos conforme o gabarito e que foram corretamente identificados como tal.

A contagem dessas métricas é realizada por meio de compreensões de lista que iteram sobre os alérgenos previstos e os alérgenos reais, realizando comparações necessárias para classificar cada alérgeno em uma das quatro categorias.

4.5.1 Levenshtein

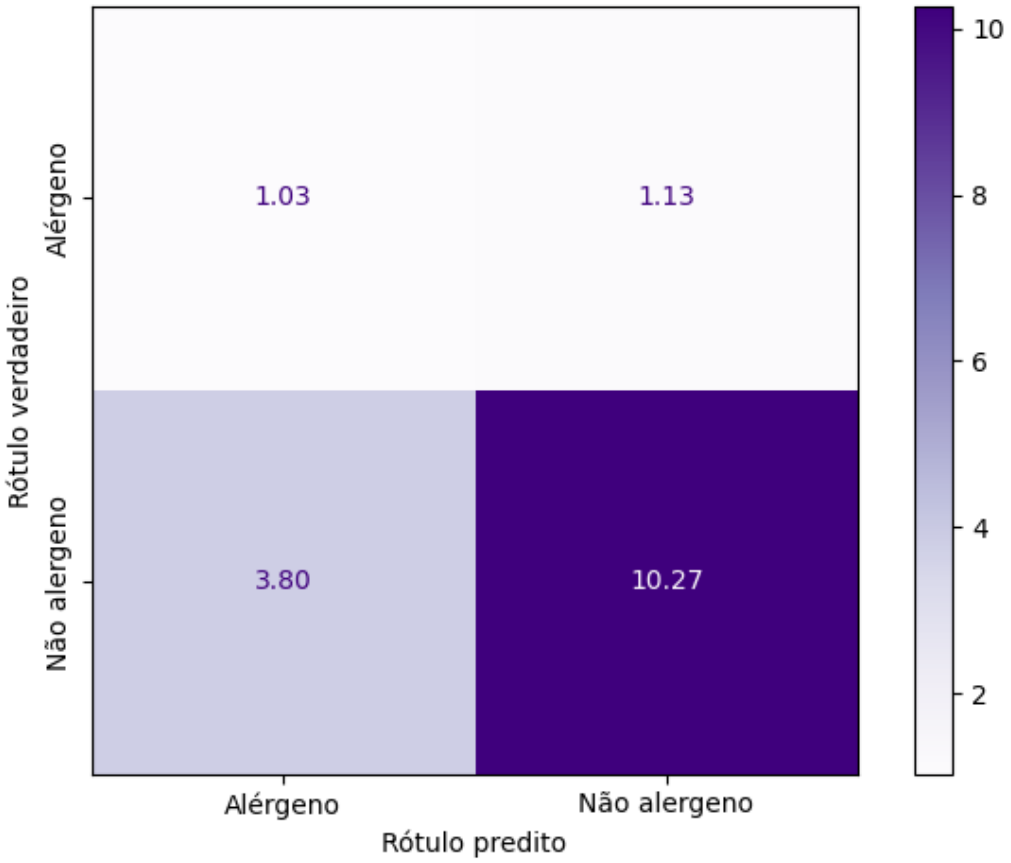

Figura 8 – Matriz de confusão para Levenshtein

Fonte: Autor do trabalho

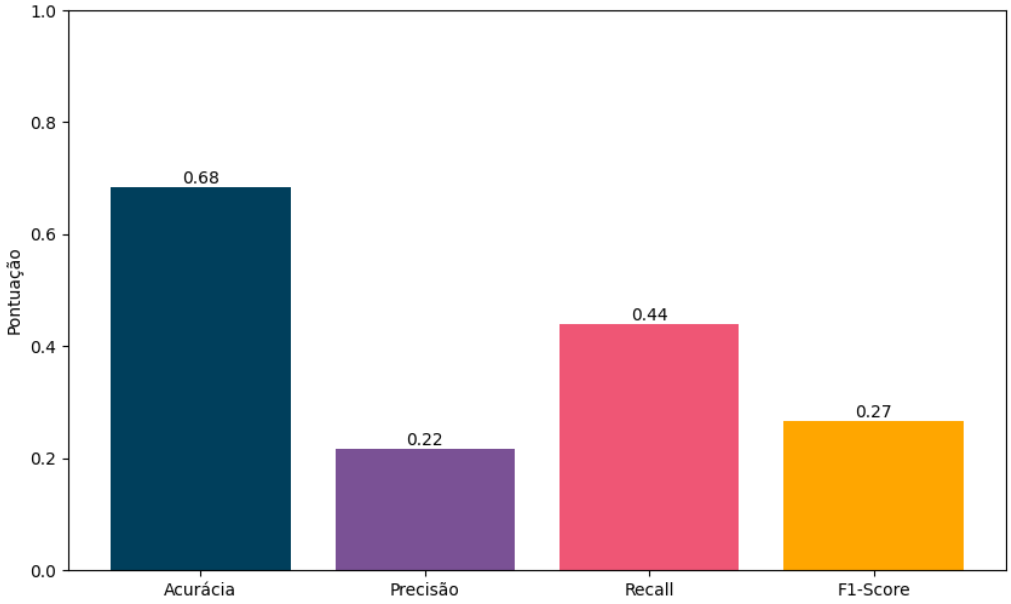

Figura 9 – Gráfico de métricas para Levenshein

Fonte: Autor do trabalho

O algoritmo Levenshtein demonstrou uma acurácia de 0,69, indicando uma capacidade moderada de fazer previsões corretas. A precisão do algoritmo foi de 0,22, sugerindo que o algoritmo tem uma quantidade significativa de falsos positivos. O recall foi de 0,44, mostrando que o algoritmo foi capaz de identificar uma porção razoável de positivos reais. O F1-Score, uma média harmônica entre precisão e recall, foi de 0,29, indicando um equilíbrio razoável entre as duas métricas.

4.5.2 Jaccard

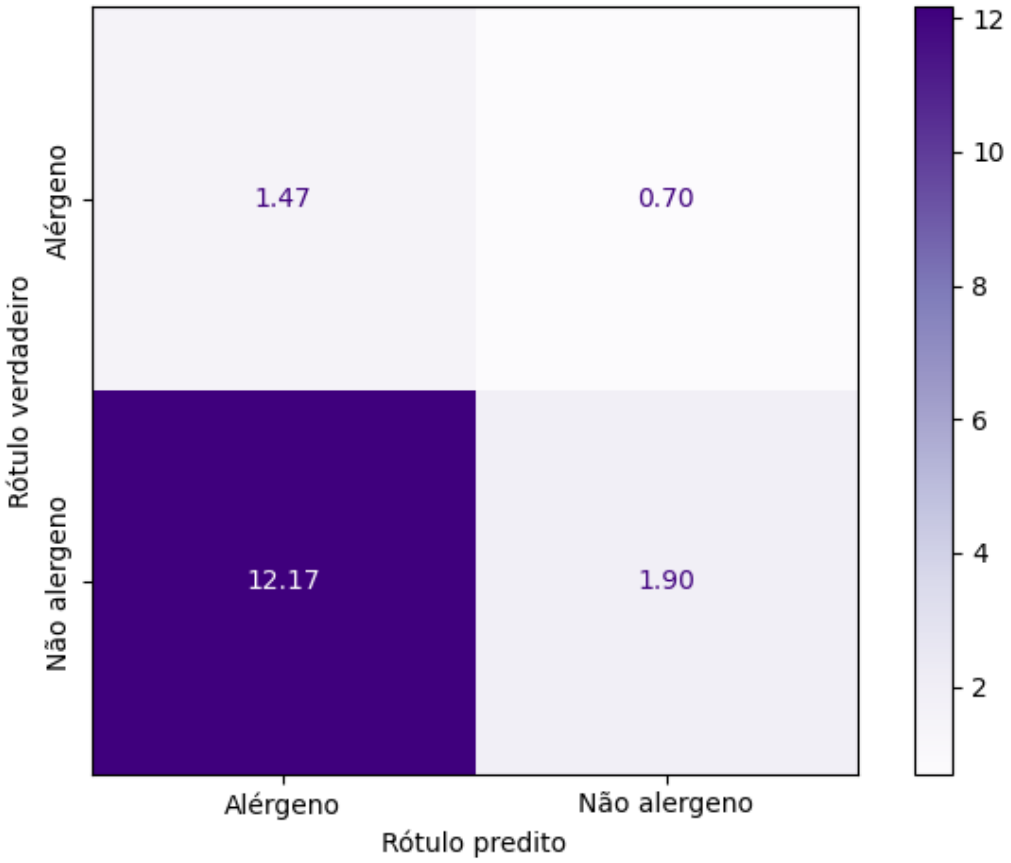

Figura 10 – Matriz de confusão para Jaccard

Fonte: Autor do trabalho

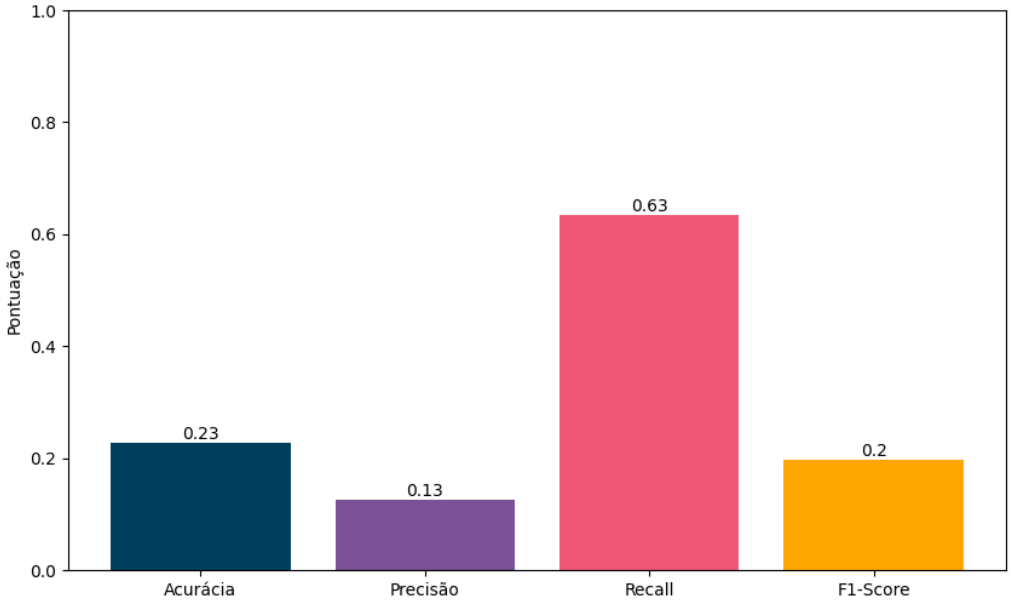

Figura 11 – Gráfico de métricas para Jaccard

Fonte: Autor do trabalho

O algoritmo Jaccard exibiu uma acurácia relativamente baixa de 0,23, o que sugere um desempenho subpar na classificação correta das instâncias. A precisão foi a mais baixa entre os algoritmos avaliados, com um valor de 0,13. No entanto, o recall foi alto (0,63), mostrando que o algoritmo é capaz de capturar uma grande proporção de positivos reais, mas com um custo de um número elevado de falsos negativos. O F1-Score foi de 0,20, o mais baixo entre os algoritmos, refletindo o desequilíbrio entre precisão e recall.

4.5.3 spaCy

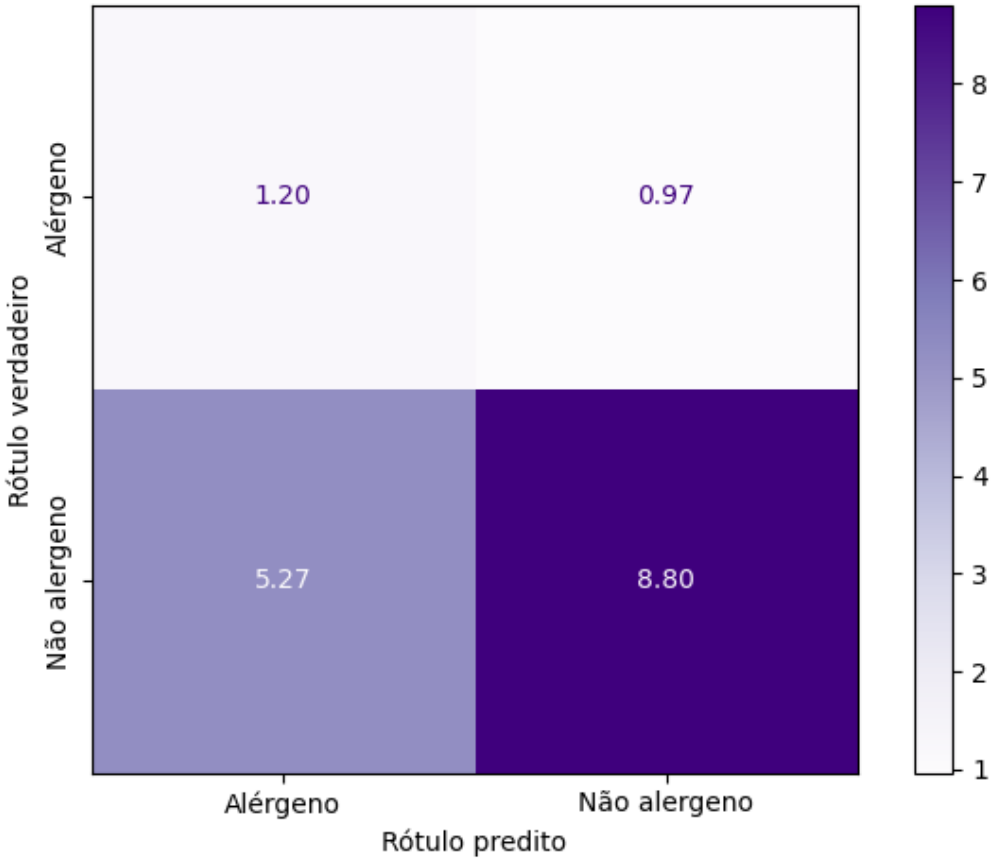

Figura 12 – Matriz de confusão para spaCy

Fonte: Autor do trabalho

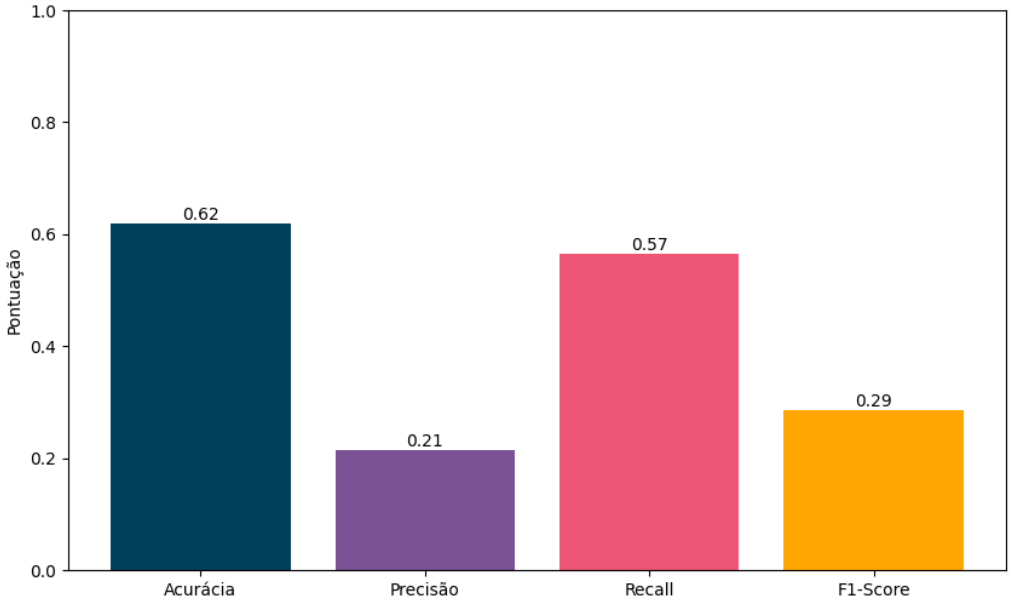

Figura 13 – Gráfico de métricas para spaCy

Fonte: Autor do trabalho

O algoritmo baseado no modelo spaCy teve uma acurácia de 0,62. A precisão foi de 0,21, e o recall foi de 0,57, indicando que o algoritmo foi relativamente bem-sucedido em identificar positivos reais. O F1-Score foi de 0,29, representando um equilíbrio aceitável entre precisão e recall.

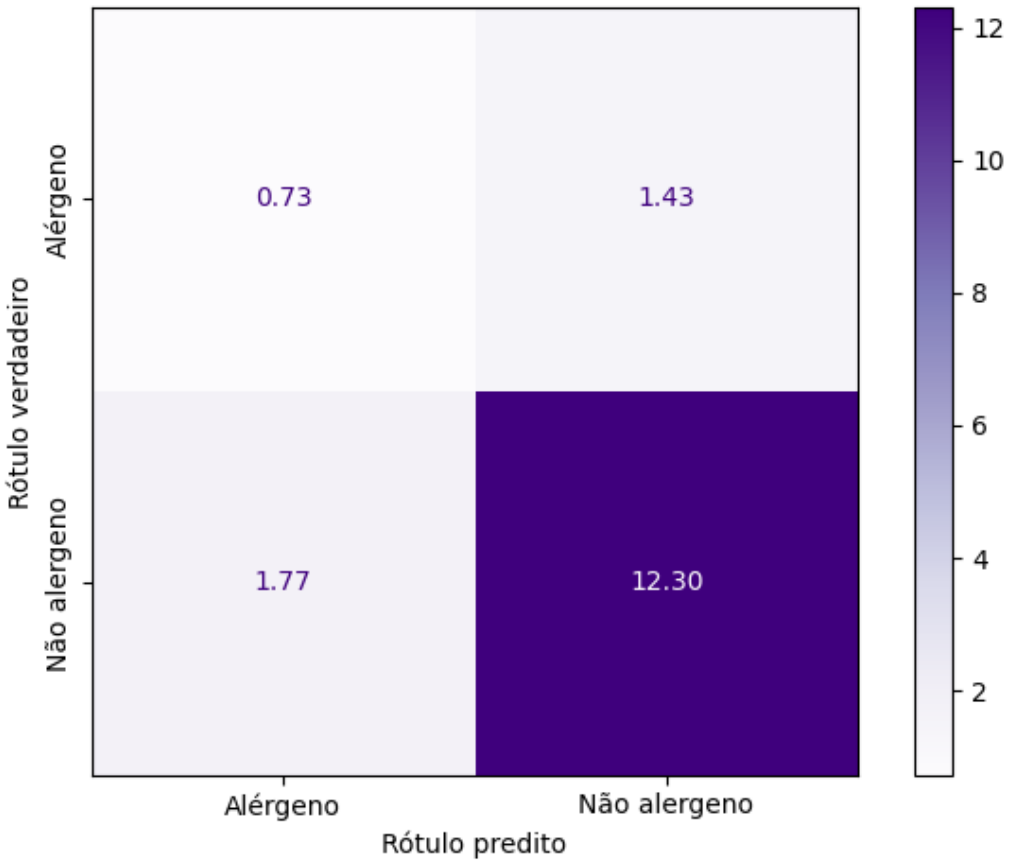

Figura 14 – Matriz de confusão para FastText

Fonte: Autor do trabalho

4.5.4 FastText

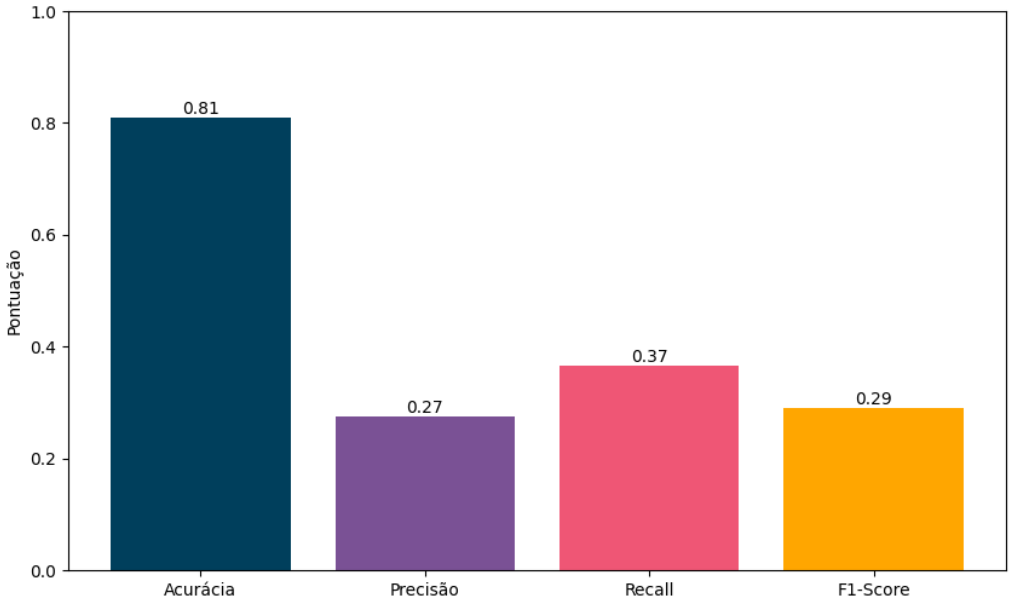

Figura 15 – Gráfico de métricas para FastText

Fonte: Autor do trabalho

FastText foi o algoritmo mais bem-sucedido em termos de acurácia, com um valor de 0,81. A precisão foi de 0,27, e o recall foi de 0,37. O F1-Score, que foi de 0,29, sugere que o FastText manteve um bom equilíbrio entre precisão e recall, sendo o algoritmo mais equilibrado neste estudo.

4.5.5 BERT

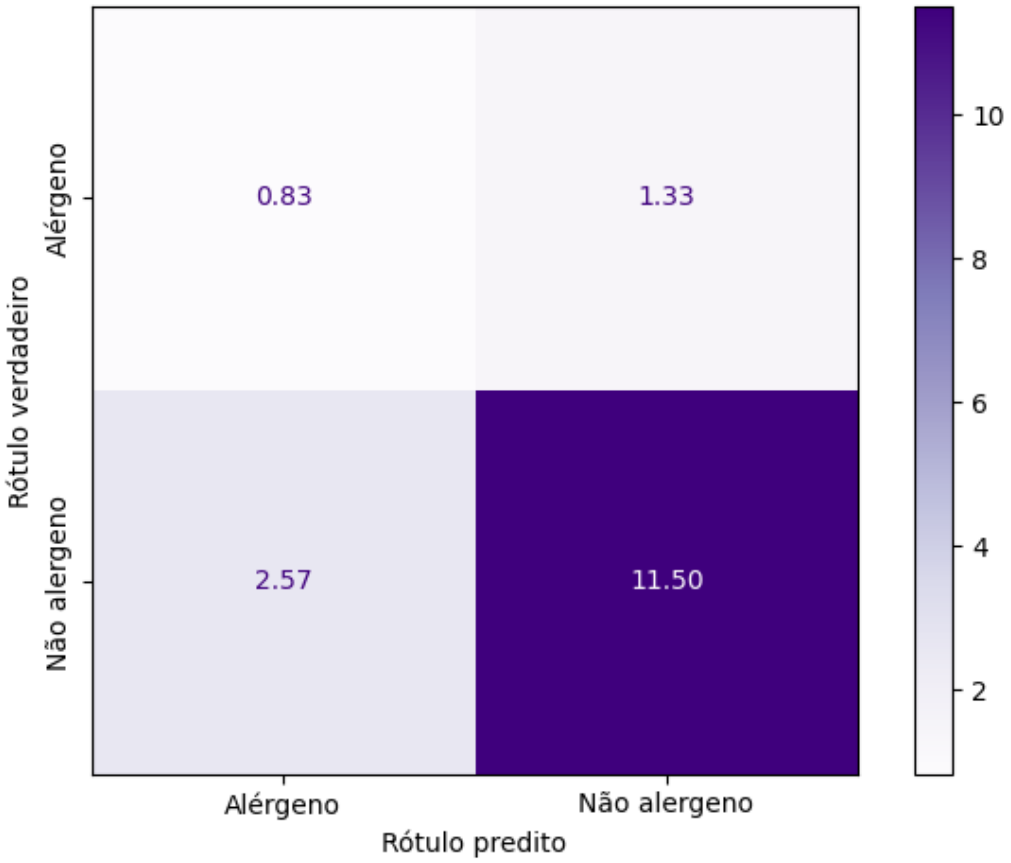

Figura 16 – Matriz de confusão para BERT

Fonte: Autor do trabalho

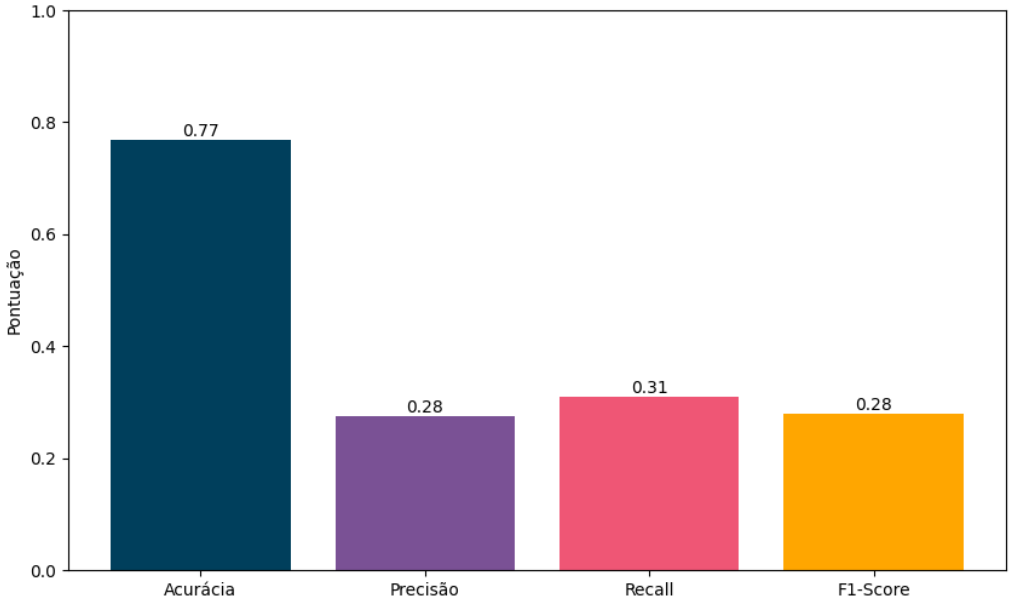

Figura 17 – Gráfico de métricas para BERT

Fonte: Autor do trabalho

O algoritmo baseado no modelo BERT apresentou uma acurácia robusta de 0,77, sendo um dos algoritmos mais precisos neste estudo. A precisão foi a mais alta entre os algoritmos, com um valor de 0,28. O recall, no entanto, foi relativamente baixo (0,31), indicando que o algoritmo pode não identificar todos os positivos reais. O F1-Score foi de 0,28, mostrando um equilíbrio razoável entre precisão e recall.

Todos algoritmos foram testados em mesmas condições. Na Tabela 5 tem-se a visão geral dos resultados da avaliação.

Tabela 5 – Comparação das métricas dos algoritmos

Métrica/Algoritmo	Levenshtein	Jaccard	spaCy	FastText	BERT
Acurácia	0.69	0.23	0.62	0.81	0.77
Precisão	0.22	0.13	0.21	0.27	0.28
Recall	0.44	0.63	0.57	0.37	0.31
F1-Score	0.27	0.20	0.29	0.29	0.28

Como comentado anteriormente, é desejado que o modelo atinja um alto recall, já que se trata de alergias alimentares que é algo crítico na saúde, porém ela não faz sentido caso não haja também uma alta precisão, indicando que apesar de poucos falsos negativos, alérgenos estão sendo corretamente detectados e que o modelo está balanceado. Por isso, na avaliação serão considerados a acurácia e F1-Score.

Com base na avaliação, o algoritmo FastText emerge como o mais promissor, exibindo a maior acurácia e um F1-Score equilibrado. BERT também mostra um desempenho sólido, com a maior precisão e um F1-Score competitivo. Jaccard, apesar de seu alto recall, tem desempenho inferior em outras métricas, sugerindo que pode não ser o mais adequado para esta tarefa. No entanto, todos os modelos apresentam desafios significativos em termos de precisão, indicando a necessidade de melhorias para reduzir os falsos negativos e melhorar a confiabilidade das predições.

Esses resultados se dão mesmo o BERT utilizando uma arquitetura mais moderna e computacionalmente complexa que captura contextos das sentenças. É importante observar que o FastText assim como o BERT realiza tokenização por subpalavras. Porém pelo fato das listas de ingredientes não possuírem contextos complexos, o BERT não pode se beneficiar de seus mecanismos de atenção, fazendo com que até arquiteturas mais simples possam se sobressair em tarefas como a apresentada neste trabalho.

5 CONCLUSÃO

A realização do projeto mostrou a viabilidade de um modelo para a detecção de alérgenos em produtos alimentícios, utilizando ontologias e aplicando técnicas de PLN diretamente na lista de ingredientes. Sendo assim, o objetivo geral foi atingido, marcando um avanço na direção de uma detecção automatizada e precisa de substâncias alergênicas.

Uma análise do estado da arte foi conduzida, proporcionando uma base conceitual para a implementação prática das técnicas de PLN escolhidas. Essas técnicas foram aplicadas em um conjunto de dados, permitindo a extração de características semânticas relevantes para a tarefa. Além disso, foi desenvolvida uma representação ontológica, capturando o conhecimento essencial sobre as relações entre diferentes alimentos. Esse conhecimento, quando integrado ao modelo proposto, potencializou a identificação racional de alérgenos. Essa integração marcou uma contribuição para o campo, possibilitando o enriquecimento contínuo de uma ontologia de alérgenos através da identificação e incorporação de novos alérgenos e seus derivados.

A avaliação e análise revelaram a eficácia do modelo em conjunto com as diversas técnicas na detecção de alérgenos, validando assim a abordagem adotada. Uma comparação sistemática foi realizada entre as técnicas, gerando percepções valiosas sobre seus respectivos desempenhos e limitações, oferecendo uma compreensão mais profunda das oportunidades e desafios na detecção de alérgenos computacionalmente.

Em reflexão aos resultados obtidos, observa-se que o modelo FastText se sobressaiu conforme o método proposto. No entanto, é importante reconhecer as limitações do método. A escassez de dados na língua portuguesa impõe um desafio, visto que muitos dos dados disponíveis estão permeados por ruídos, exigindo uma fase de preparação de dados mais intensiva para mitigar impactos negativos tanto na evolução da ontologia quanto na precisão da detecção de alérgenos. Além disso, o modelo demonstra maior eficácia quando os ingredientes mencionam explicitamente os alérgenos inseridos inicialmente na ontologia. Quando os ingredientes são descritos de forma indireta ou com terminologias atípicas, o modelo enfrenta obstáculos significativos. Para aprimorar a eficácia do modelo, uma abordagem prática ainda a ser explorada é a classificação por comitê. Esse método consiste na combinação de múltiplos dos algoritmos e técnicas utilizadas, permitindo a sinergia de suas individualidades e culminando em uma classificação mais refinada e rigorosa. Diante desses aspectos, vislumbra-se um caminho para trabalhos futuros. A publicação dos resultados em plataformas de Web of Data pode ampliar a visibilidade e a aplicabilidade do modelo, assim como a integração com outras ontologias poderia enriquecer ainda mais o conhecimento representado. Ademais, há a possibilidade de adaptar e incorporar o modelo em aplicativos de software que, por meio de OCR, podem detectar alérgenos em rótulos de produtos alimentícios em tempo real, oferecendo uma ferramenta valiosa para consumidores com restrições alimentares.

REFERÊNCIAS

AHAD, Md Atiqur Rahman; ANTAR, Anindya das; AHMED, Masud. Performance Evaluation in Activity Classification: factors to consider. **Iot Sensor-Based Activity Recognition**, [S.L.], p. 133-147, 31 jul. 2020. Springer International Publishing.

AMITH, Muhammad; ONYE, Chidinma; LEDOUX, Tracey; XIONG, Grace; TAO, Cui. The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications. **Bmc Medical Informatics And Decision Making**, [S.L.], v. 21, n. 7, p. 275-290, 9 nov. 2021. Springer Science and Business Media LLC.

BOJANOWSKI, Piotr, et al. **Enriching Word Vectors with Subword Information**. 2016.

CAMBRIA, Erik; WHITE, Bebo. Jumping NLP Curves: A Review of Natural Language Processing Research **IEEE Computational Intelligence Magazine p.** 48-57 apr. 2014. Disponível em: https://ieeexplore.ieee.org/document/6786458. Acesso em: 12 jul. 2022.

CASTELLS-RUFAS, David. GPU acceleration of Levenshtein distance computation between long strings. **Parallel Computing**, [S.L.], v. 116, p. 103019, jul. 2023. Elsevier BV. http://dx.doi.org/10.1016/j.parco.2023.103019.

CHOWDHARY, K. R. **Fundamentals of Artificial Intelligence**. 1 ed. Jodhpur: Springer, 2020.

CLUNIS, Julaine. Designing an ontology for managing the diets of hypertensive individuals. **International Journal On Digital Libraries**, [S.L.], v. 20, n. 3, p. 269-284, 22 out. 2018. Springer Science and Business Media LLC.

DEVLIN, Jacob, et al. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. 2018.

GLIMM, Birte; HORROCKS, Ian; MOTIK, Boris; STOILOS, Giorgos; WANG, Zhe. HermiT: an owl 2 reasoner. **Journal Of Automated Reasoning**, [S.L.], v. 53, n. 3, p. 245-269, 23 maio 2014. Springer Science and Business Media LLC.

GOLDENBERG, Elazar; KOCIUMAKA, Tomasz; KRAUTHGAMER, Robert; SAHA, Barna. An Algorithmic Bridge Between Hamming and Levenshtein Distances. **Schloss Dagstuhl - Leibniz-Zentrum Für Informatik**, [S.L.], v. 251, n. 58, p. 1-23, fev. 2023. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

GRUBER, Thomas Robert. Toward principles for the design of ontologies used for knowledge sharing? **International Journal Of Human-Computer Studies.** Palo Alto, p. 907-928. nov. 1995. Disponível em: https://www.sciencedirect.com/science/article/pii/S1071581985710816. Acesso em: 05 fev. 2022.
HALDAR, Rishin; DEBAJYOTI Mukhopadhyay. Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach. 2011.

HAUSSMANN, Steven; SENEVIRATNE, Oshani; CHEN, Yu; NE'EMAN, Yarden; CODELLA, James; CHEN, Ching-Hua; MCGUINNESS, Deborah L.; ZAKI, Mohammed J.. FoodKG: a semantics-driven knowledge graph for food recommendation. **Lecture Notes In Computer Science**, [S.L.], p. 146-162, 2019. Springer International Publishing.

HEYDARIAN, Mohammadreza; DOYLE, Thomas E.; SAMAVI, Reza. MLCM: multi-label confusion matrix. **Ieee Access**, [S.L.], v. 10, p. 19083-19095, 2022. Institute of Electrical and Electronics Engineers (IEEE).

KHURANA, Diksha; KOLI, Aditya; KHATTER, Kiran; SINGH, Sukhdev. Natural language processing: state of the art, current trends and challenges. **Multimedia Tools And Applications**, [S.L.], v. 82, n. 3, p. 3713-3744, 14 jul. 2022. Springer Science and Business Media LLC.
LACSON, Ronilda; LONG, William. Natural Language Processing of Spoken Diet Records (SDRs). **Amia Annu Symp Proc.** Cambridge, p. 454-458.2006.

LAURIOLA, Ivano; LAVELLI, Alberto; AIOLLI, Fabio. An introduction to Deep Learning in Natural Language Processing: models, techniques, and tools. **Neurocomputing**, [S.L.], v. 470, p. 443-456, jan. 2022. Elsevier BV.

LIMA, Júnio César de; CARVALHO, Cedric Luiz de. **Ontologias - OWL (Web Ontology Language)**. Goiânia: Universidade Federal de Goiás, 2005.

LUGER, George. **Inteligência Artificial**. Tradução Daniel Vieira. 6 ed. São Paulo: Pearson, 2013.

MARTINEZ-PLUMED, Fernando; CONTRERAS-OCHANDO, Lidia; FERRI, Cesar; HERNANDEZ-ORALLO, Jose; KULL, Meelis; LACHICHE, Nicolas; RAMIREZ-QUINTANA, Maria Jose; FLACH, Peter. CRISP-DM Twenty Years Later: from data mining processes to data science trajectories. **Ieee Transactions On Knowledge And Data Engineering**, [S.L.], v. 33, n. 8, p. 3048-3061, 1 ago. 2021. Institute of Electrical

and Electronics Engineers (IEEE).

MARTINS, Túlio Brandão Soares; REIS, Julio Cesar dos. **Addressing Inconsistencies in the DBPedia Evolution**. Campinas: Universidade Estadual de Campinas, 2019.

MINISTÉRIO DA SAÚDE. Constituição (2015). Resolução da Diretoria Colegiada nº 26, de 02 de julho de 2015. Brasília.
MINISTÉRIO DA SAÚDE. Constituição (2017). Resolução nº 1112, de 02 de maio de 2017. Brasília.

NARDI, Daniele; BRACHMAN, Ronald J. An Introduction to Description Logics. In: BAADER, Franz *et al.* **The Description Logic Handbook**: theory, implementation, and applications. Cambridge: Cambridge University Press, 2003. p. 1-44. Disponível em: https://www.cambridge.org/core/books/abs/description-logic-handbook/an-introduction-to-description-logics/45368068B25094C3D939E3542B00B2FA. Acesso em: 05 fev. 2022.

NIWATTANAKUL, Suphakit et al. **Using of Jaccard Coefficient for Keywords Similarity.** 2013.

PARSIA, Bijan; MATENTZOGLU, Nicolas; GONÇALVES, Rafael S.; GLIMM, Birte; STEIGMILLER, Andreas. The OWL Reasoner Evaluation (ORE) 2015 Competition Report. **Journal Of Automated Reasoning**, [S.L.], v. 59, n. 4, p. 455-482, 21 fev. 2017. Springer Science and Business Media LLC.

PILEHVAR, Mohammad Taher; CAMACHO-COLLADOS, Jose. **Embeddings in Natural Language Processing**: theory and advances in vector representation of meaning. [S. L.]: Springer Cham, 2021.

RAHUTOMO, Faisal; KITASUKA, Teruaki; ARITSUGI, Masayoshi. **Semantic Cosine Similarity.** 2012**.**

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. Tradução Regina Célia Simille. 3 ed. Rio de Janeiro: Elsevier, 2013.

RUUSKA, Salla; HÄMÄLÄINEN, Wilhelmiina; KAJAVA, Sari; MUGHAL, Mikaela; MATILAINEN, Pekka; MONONEN, Jaakko. Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. **Behavioural Processes**, [S.L.], v. 148, p. 56-62, mar. 2018. Elsevier BV.

SAWICKI, Jan; GANZHA, Maria; PAPRZYCKI, Marcin. The State of the Art of Natural Language Processing—A Systematic Automated Review of NLP Literature Using NLP Techniques. **Data Intelligence**, [S.L.], v. 5, n. 3, p. 707-749, 2023. MIT Press.

SHEARER**,** Rob; MOTIK, Boris; HORROCKS, Ian. HermiT: A Highly-Efficient OWL Reasoner. 2008.

SNAE, Chakkrit; BRUCKNER, Michael. FOODS: a food-oriented ontology-driven system. **2008 2Nd Ieee International Conference On Digital Ecosystems And Technologies**, Phitsanuloke, v. 2, n. 2, p. 168-176, fev. 2008. IEEE.

SOKOLOVA, Marina; LAPALME, Guy. A systematic analysis of performance measures for classification tasks. **Information Processing & Management**, [S.L.], v. 45, n. 4, p. 427-437, jul. 2009. Elsevier BV.

SOUZA, Fábio Capuano de. **BERTimbau: modelos BERT pré-treinados para Português Brasileiro**. 2020. 62 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, na Área de Engenharia de Computação, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2020.

TURING, Alan Mathison. Computing Machinery and Intelligence. **Mind.** Oxford, p. 433-460. out. 1950. Disponível em: https://academic.oup.com/mind/article/LIX/236/433/986238. Acesso em: 05 fev. 2022.

VASWANI, Ashish, et al. **Attention Is All You Need**. 2017.

VAZ, Arthur Lamblet. **Aplicação de Deep Learning para o reconhecimento de tabelas nutricionais e de ingredientes nos produtos alimentícios**. 2018. 40 f. Monografia (Especialização) - Curso de Business Intelligence, Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2018.

WANG, Jiapeng; DONG, Yihong. Measurement of Text Similarity: a survey. **Information**, [S.L.], v. 11, n. 9, p. 421, 31 ago. 2020. MDPI AG.

WILKINSON, Sophie L.. Deconstructing Food Allergies. **Chemical & Engineering News Archive**, [S.L.], v. 76, n. 36, p. 38-40, 7 set. 1998. American Chemical Society (ACS).

YOUNG, Julio Christian; RUSLI, Andre. “Review and Visualization of Facebook’s FastText Pretrained Word Vector Model.” 2019 International Conference on Engineering, Science, and Industrial Applications (ICESI), IEEE, 2019, pp. 1–6.

ZHANG, Jinqu; QIAN, Lang; WANG, Shu; ZHU, Yunqiang; GAO, Zhenji; YU, Hailong; LI, Weirong. A Levenshtein distance-based method for word segmentation in corpus augmentation of geoscience texts. **Annals Of Gis**, [S.L.], v. 29, n. 2, p. 293-306, 10 jan. 2023. Informa UK Limited.

APÊNDICE A – ARTIGO

Uso de processamento de linguagem natural para detecção de alérgenos em alimentos a partir da lista de ingredientes

Christian Aurich Zanettini Martins[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

christian.aurichzm@gmail.com

Abstract. *This work explores the use of natural language processing and ontologies to help interpret food labels, a vital measure for nutritional monitoring and prevention of food allergy symptoms. Given the increasing prevalence of these allergies, the study developed an automated model for analyzing and detecting allergens in ingredient lists. Experiments were carried out to identify the most efficient NLP techniques in this task. The results contribute to improving the model and provide perspectives for future research.*

Resumo. *Este trabalho explora o uso de processamento de linguagem natural e ontologias para auxiliar a interpretação de rótulos alimentares, uma medida vital para o acompanhamento nutricional e prevenção dos sintomas das alergias alimentares. Dada a crescente prevalência dessas alergias, o estudo desenvolveu um modelo automatizado para análise e detecção de alérgenos em listas de ingredientes. Experimentos foram realizados para identificar as técnicas mais eficientes de PLN nesta tarefa. Os resultados contribuem para o aperfeiçoamento do modelo e fornecem perspectivas para futuras pesquisas.*

1. Introdução

As técnicas de PLN, juntamente com ontologias, têm sido aplicadas para resolver uma variedade de problemas. No contexto deste trabalho, elas são empregadas para abordar um problema na saúde populacional: a detecção de alérgenos em listas de ingredientes de alimentos. Essa identificação de alérgenos é fundamental para a prevenção de reações adversas em pessoas com essa sensibilidade. Os rótulos e listas de ingredientes são fontes ricas de informações sobre os alimentos, e muitas vezes estão repletas de detalhes e conhecimento implícitos, tornando a tarefa de identificação manual de alérgenos exaustiva. Dada as certezas e possibilidade de explicitação acerca da relação entre alergias e alimentos, este estudo visa desenvolver um modelo inteligente dedicado a automatizar e otimizar essa tarefa.

2. Objetivos

Nas subseções abaixo estão descritos o objetivo geral e os objetivos específicos do trabalho.

2.1. Objetivo geral

Desenvolver um modelo para detecção de substâncias relacionadas a alergias comuns em alimentos, através da aplicação de técnicas de processamento de linguagem natural no texto de listagem de seus ingredientes.

2.2. Objetivos específicos

- Análise do estado da arte em PLN
- Aplicação de PLN para extração semântica
- Representação ontológica de alérgenos
- Integração de ontologia e semântica
- Avaliação da eficácia do modelo
- Comparação de técnicas de PLN

3. Método de pesquisa

Este trabalho é conduzido por um conjunto estruturado de etapas as quais não seguem estritamente a um procedimento metodológico. Com objetivos definidos, o processo foi parcialmente inspirado por metodologias consolidadas na área como o CRISP-DM. No entanto, conforme discutido por Martinez-Plumed et al. (2021), a natureza exploratória desta pesquisa exige uma abordagem metodológica mais flexível e adaptável aos objetivos específicos do estudo. A seguir tem-se as etapas que compõem o método.

3.1. Revisão bibliográfica

Uma revisão da literatura foi realizada, englobando livros e trabalhos acadêmicos da área de estudo e de domínio do problema. Essa etapa foi crucial para aquisição de um entendimento do domínio facilitando a execução das etapas subsequentes e construção da fundamentação teórica. Também foi buscado identificar métodos e técnicas relevantes. Esta etapa, além de um levantamento da literatura, foi uma análise crítica que ajudou a refinar o objetivo da pesquisa e a traçar um procedimento metodológico adequado. Também foram procuradas lacunas e oportunidades para inovação no desenvolvimento do trabalho.

3.2. Obtenção dos dados

Os dados foram pesquisados e coletados em conjuntos de dados públicos disponíveis na *web*, focando em produtos alimentícios e seus ingredientes. A seleção do conjunto foi baseada na relevância, volume e qualidade para assegurar a robustez na fase de modelagem.

3.3. Preparação dos dados

Nesta fase, técnicas de pré-processamento foram aplicadas aos dados coletados para otimizá-los, garantindo que estejam em um formato adequado para alimentar o modelo e favorecer a obtenção de resultados precisos e confiáveis.

3.4. Experimentações

O modelo é mais do que uma implementação técnica; é uma representação que deve ser submetida a testes para validar sua eficácia e precisão.

3.5. Escolha das técnicas

A partir das experimentações, foi possível escolher as técnicas e parâmetros mais apropriados para serem utilizadas no desenvolvimento do trabalho.

3.6. Construção do modelo

O modelo foi construído integrando uma base de conhecimentos sobre alimentos utilizando ontologias, e combinando com as características extraídas durante a preparação dos dados. Este modelo visa a identificação de alérgenos nos textos dos produtos alimentícios e é utilizado em conjunto com uma variedade de algoritmos e modelos distintos com propósito comparativo.

3.7. Avaliação e análise do modelo

Foi realizada a descrição do processo de avaliação dos resultados obtidos, incluindo a análise crítica das métricas e resultados, e como eles são comparados não apenas entre as diferentes técnicas utilizadas neste estudo, mas também em relação às práticas estabelecidas no estado da arte. Essa abordagem fundamenta a aplicação do modelo e das técnicas empregadas para demais propósitos relacionados.

4. Revisão bibliográfica

As seguintes áreas foram revisadas e fundamentadas para elaboração do presente trabalho:

- Alergias alimentares
- Ontologias
- Processamento de linguagem natural
- Avaliação de modelos

5. Trabalhos relacionados

Durante a revisão bibliográfica foram identificados estudos relevantes que se alinham com a temática e as metodologias empregadas neste trabalho, utilizando procedimentos e ferramentas análogas para alcançar objetivos convergentes.

Para estabelecer uma conexão clara entre este estudo e trabalhos anteriores, foram adotados alguns critérios de avaliação. Entre eles, destacam-se a criação de ontologias e a implementação de processamento de linguagem natural no contexto da alimentação e nutrição. Este seção lista esses estudos que fornecem uma contextualização do presente trabalho.

1. Designing an Ontology for Managing the Diets of Hypertensive Individuals
2. FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation
3. The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications
4. FOODS: A Food-Oriented Ontology-Driven System
5. Aplicação de Deep Learning para o reconhecimento de tabelas nutricionais e de ingredientes nos produtos alimentícios
6. Natural Language Processing of Spoken Diet Records (SDRs)

6. Experimentos

Durante a fase experimental, foram exploradas diversas outras técnicas de PLN para compor a transformação dos dados e aprimorar a extração de *features*. Embora não tenham sido incorporadas à versão final do modelo, elas apresentam um potencial significativo para seu refinamento. Também foram realizados experimentos com algoritmos e modelos de aprendizado de máquina que calculam similaridade entre *strings*, tendo sido selecionados os que demonstraram maior viabilidade de implementação e que obtiveram resultados preliminares promissores.

7. Desenvolvimento

A complexidade inerente à essa tarefa de detecção de alérgenos, dada a diversidade e ambiguidade na forma com que os alérgenos podem ser apresentados nos rótulos, motivou a busca por uma abordagem computacional inteligente para realizá-la. Essa abordagem a ser apresentada é implementada de maneira simbólica. Neste contexto, as ontologias desempenham um papel crucial, funcionando como mecanismo de raciocínio do sistema. E além de poder indicar quais são os alérgenos conhecidos, também auxilia ativamente na desambiguação dos termos. Esse processo é vital para garantir que o sistema identifique corretamente os alérgenos, mesmo quando estes são listados de maneiras diferentes nos rótulos dos produtos, proporcionando assim uma camada adicional de precisão e confiabilidade à detecção de alérgenos realizada.

Para a tarefa de detecção de alérgenos é crucial entender a estrutura das listas de ingredientes e considerar seus contextos. No entanto, é importante salientar que o contexto encontrado em listas de ingredientes é intrinsecamente diferente do encontrado em textos narrativos, diálogos ou conversas. Nas listas de ingredientes o contexto é geralmente mais simples e estruturado. Ele pode incluir hierarquização de ingredientes, onde ingredientes principais podem ter sub-ingredientes listados entre parênteses, ou a inclusão de modificadores que especificam o estado ou a forma de um ingrediente. Esse tipo de contexto é mais direto e menos propenso a ambiguidades.

A proposta de solução a ser desenvolvida segue o procedimento metodológico descrito no trabalho. Uma visão geral dela pode ser vista na Figura 1.

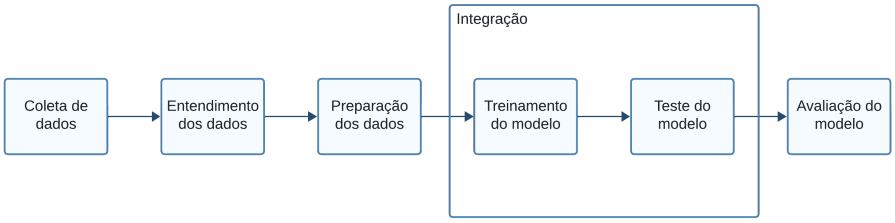

Figura 1. Fluxograma do desenvolvimento
Fonte: Autor do trabalho

7.1. Coleta de dados

O conjunto de dados empregado no projeto foi obtido a partir do site oficial do Open Food Facts, um banco de dados público e aberto de produtos alimentícios. Esse repositório é notório por seu volume significativo de dados, que são contribuídos por uma comunidade ativa de usuários, pesquisadores e profissionais das áreas relacionadas em todo o mundo. No momento de escrita deste trabalho, a plataforma conta com aproximadamente 3 milhões de produtos registrados.

Para as necessidades específicas deste projeto, foi conveniente buscar apenas os dados que estivessem em conformidade com o contexto linguístico e geográfico de interesse. Esse processo foi complementado com uma filtragem diretamente através da

API fornecida pelo Open Food Facts, a qual permite aos usuários acessar de maneira pragmática.

O processo de filtragem foi direcionado a obter produtos alimentícios cuja informações estivessem disponíveis no idioma português. O conjunto de dados resultante foi adquirido no formato CSV, proporcionando uma base estruturada com cerca de 15 mil produtos para a condução das experimentações planejadas no âmbito deste trabalho.

7.2. Preparação dos dados

O Open Food Facts sendo um banco de dados aberto, permite que usuários registrados na plataforma insiram dados. Essa inserção pode ocorrer por meio de submissão de texto ou de imagens de rótulos de alimentos. No caso de imagens, são empregados métodos de reconhecimento óptico de caracteres para transcrever automaticamente as informações contidas nos rótulos. Contudo, esse processo está sujeito a falhas, sejam elas humanas ou oriundas do software utilizado. Dessa forma, é imprescindível a realização de um pré-processamento dos dados antes de sua utilização. Os dados obtidos no formato CSV foram manipulados utilizando a biblioteca pandas. Inicialmente foi realizada a divisão do conjunto de dados em dois grupos: treinamento e teste. O conjunto de treinamento tem como função primordial de enriquecer a ontologia, populando-a com novos indivíduos derivados dos alérgenos iniciais. Esse conjunto é a base sobre a qual o modelo é construído e refinado.

Subsequentemente foi realizada a limpeza dos dados através das seguintes operações:

- Eliminar instâncias com listas de ingredientes nulas;
- Transformação de todo o texto para letras minúsculas;
- Remoção de espaços em branco nas extremidades das strings;
- Remoção de diacríticos
- Utilização de expressões regulares para remover caracteres especiais e sequências de vírgulas e espaços

No estágio final de preparação de dados, foi procedida a extração de *features*, empregando técnicas de processamento de linguagem natural para isolar termos relevantes nas listas de ingredientes. Para essa tarefa foi elaborada uma expressão regular para segmentar os ingredientes, utilizando vírgulas e a conjunção ‘e’ como delimitadores.

A Figura 2 demonstra de forma esquematizada as etapas da preparação de dados.

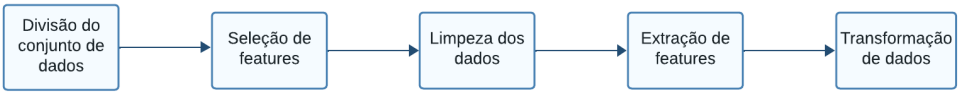

Figura 2. Fluxograma da etapa de preparação dos dados
Fonte: Autor do trabalho

A Tabela 1 ilustra exemplos práticos dessa transformação, mostrando como as listas originais de ingredientes são processadas e convertidas

Tabela 1. Exemplo de preparação de dados para o modelo

Lista de ingredientes original	**Feature resultante da preparação de dados**
Tomate, cebola, proteína texturizada de soja, sal, amido modificado, extrata de adura alho, salsa, acidulante ácido lático e aroma idêntico ao natural de bolonhesa.	[’tomate’, ’cebola’, ’proteina texturizada de soja’, ’sal’, ’amido modificado’, ’extrata de adura alho’, ’salsa’, ’acidulante acido latico’, ’aroma identico ao natural de bolonhesa’]
Aveia em flocos, flocos de cevada, açúcar demerara, extrato de malte, fibra de trigo, flocos de arroz, açúcar mascavo, maltodextrina, coco seco, flocos de milho, polpa de banana, cacau em pó, farinha de soja, gergelim, flocos de soja, óleo de girassol, linhaça, mel, aroma idêntico ao natural de chocolate maltado e antioxidante tocoferol (natural).	[’aveia em flocos’, ’flocos de cevada’, ’acucar demerara’, ’extrato de malte’, ’fibra de trigo’, ’flocos de arroz’, ’acucar mascavo’, ’maltodextrina’, ’coco seco’, ’flocos de milho’, ’polpa de banana’, ’cacau em po’, ’farinha de soja’, ’gergelim’, ’flocos de soja’, ’oleo de girassol’, ’linhaca’, ’mel’, ’aroma identico ao natural de chocolate maltado’, ’antioxidante tocoferol natural’]

7.3. Ontologia

Foi construída uma ontologia para representar o domínio de alérgenos alimentares. Ela é codificada seguindo a especificação RDF e incorporando o vocabulário e construções semânticas oferecidas pela OWL.

A ontologia é composta por classes, propriedades de objetos e indivíduos que representam conceitos relacionados a alimentos e alérgenos alimentares.

A Figura 3 apresenta a visão geral da ontologia.

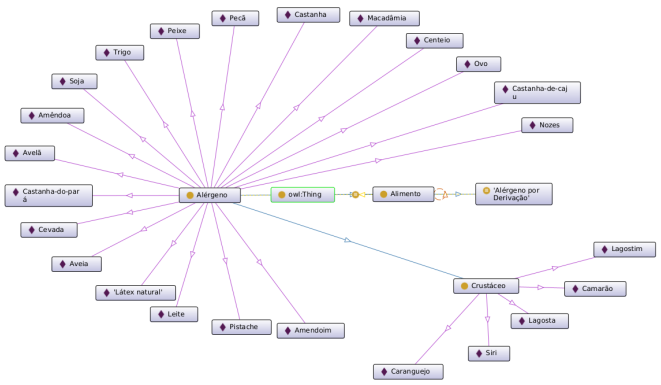

Figura 3. Visão geral da ontologia
Fonte: Autor do trabalho

7.3.1. Classes

As classes que compõe a ontologia são:

- Alimento: Classe base que representa um superconjunto de todos os alimentos
- Alergeno: Subclasse de Alimento, denotando os alérgenos alimentares
- AlergenoPorDerivacao: Subclasse de Alimento, representando alimentos que são alérgenos devido à sua derivação de outros alérgenos
- Crustaceo: Subclasse de Alergeno, indicando que alimentos de origem animal do subfilo Crustáceo são alérgenos

7.3.2. Propriedades de objetos

eDerivadoDe é uma propriedade que relaciona um alimento a outro do qual é derivado, com domínio e alcance definidos pela classe Alimento.

7.3.3. Indivíduos

Os indivíduos iniciais da ontologia são baseados na resolução RDC 26/2015 da Anvisa, que cataloga os principais alimentos causadores de alergias.

1. Trigo, centeio, cevada, aveia e suas estirpes hibridizadas.
2. Crustáceos.
3. Ovos.
4. Peixes.
5. Amendoim.
6. Soja.
7. Leites de todas as espécies de animais mamíferos.
8. Amêndoa (Prunus dulcis, sin.: Prunus amygdalus, Amygdalus communis L.).
9. Avelãs (Corylus spp.).
10. Castanha-de-caju (Anacardium occidentale).
11. Castanha-do-brasil ou castanha-do-pará (Bertholletia excelsa).
12. Macadâmias (Macadamia spp.).
13. Nozes (Juglans spp.).
14. Pecãs (Carya spp.).
15. Pistaches (Pistacia spp.).
16. Pinoli (Pinus spp.).
17. Castanhas (Castanea spp.).
18. Látex natural.

É possível perceber que a lista conta com algumas ambiguidades que precisaram ser abstraídas para simplificação do modelo. É o caso da omissão dos nomes científicos dos alimentos. Já algumas características foram incorporadas como os Crustáceos, que sendo um subfilo do reino animal, engloba diversas espécies que podem aparecer isoladamente nas listas de ingredientes.

Sendo assim, indivíduos resultantes após a análise foram:

- Amendoim
- Amêndoa
- Aveia
- Avelã
- Camarão
- Caranguejo
- Castanha
- Castanha-de-caju
- Castanha-do-brasil
- Castanha-do-pará
- Centeio
- Cevada
- Lagosta
- Lagostim
- Leite
- Látex natural
- Macadâmia
- Nozes
- Ovo
- Pecã
- Peixe
- Pistache
- Siri
- Soja
- Trigo

7.3.4. Relações semânticas

A classe AlergenoPorDerivacao é definida como qualquer instância de Alimento que é derivado de um Alergeno.

7.3.5. Anotações e rótulos

Por convenção, cada classe e indivíduo na ontologia é anotado com um rótulo que facilita consultas e oferece flexibilidades.

7.3.6. Equivalência semântica

O indivíduo Castanha-do-brasil é declarado como equivalente de Castanha-do-pará através do predicado sameAs pois ambos são nomes conhecidos para o mesmo alimento e podem aparecer nos rótulos dos alimentos intercambiavelmente.

7.3.7. Inferências

Todos os indivíduos que possuam a propriedade de objeto descrevendo que o mesmo é um derivado de um indivíduo do tipo Alergeno, é inferido como do tipo AlergenoPorDerivacao.

7.4. Integração

As variações de algoritmo utilizados constam na Tabela 2.

Tabela 2. Comparação computacional dos algoritmos

Algoritmo	Complexidade	Necessita de modelo treinado	Tipo de embedding
Levenshtein	$O(n \cdot m)$	Não	-
Jaccard	$O(n + m)$	Não	-
spaCy	$O(d)$	Sim	Não contextual
FastText	$O(w \cdot d + d)$	Sim	Não contextual
BERT	$O(w \cdot d^2 + d)$	Sim	Contextual

'n': tamanho da string do alérgeno;
'm': tamanho da string do ingrediente;
'd': dimensão dos vetores de embedding;
'w': número de palavras de uma sentença.

7.4.1. Treinamento do modelo

Optou-se por realizar uma incorporação prévia de alérgenos na ontologia baseado na heurística de que ingredientes cujas strings que contém nomes de alérgenos já presentes na ontologia são inseridos. Esse procedimento amplia o conhecimento inicial sobre o ambiente e o domínio do problema, proporcionando uma base mais robusta para a identificação e análise de alérgenos.

Contudo, a simplicidade da heurística pode ocasionar em adições equivocadas devido a nuances não capturadas durante a limpeza de dados. Também há casos em que um sistema com uma ontologia previamente populada a partir de uma fonte de dados poderá ter dificuldade de identificar, como é o caso do ingrediente "leite de arroz", pois como é descrito na tabela da ANVISA, todos os leites alergênicos são os derivados de animais mamíferos, além dos derivados de outros alergênicos como a soja.

7.4.2. Teste do modelo

O raciocinador utilizado no modelo foi o HermiT. Ele foi escolhido por ter conformidade com a OWL 2 Direct Semantics, garantindo que está de acordo com os padrões atuais e bem aceitos da comunidade. Como exposto por Shearer et al. (2008), ele também conta com diversas otimizações de desempenho e mostra ser capaz de classificar ontologias que outros raciocinadores baseados em lógica descritiva não conseguem, além de ser tão eficiente em termos de desempenho quanto. É o raciocinador padrão da biblioteca owlready2, que cria uma interface entre programas Python e ontologias OWL. O HermiT é

compatível com SPARQL, a linguagem de consultas utilizada neste trabalho para integrar a ontologia ao sistema.

Sendo assim, foi desejado que após o raciocinador fosse ligado, o resultado das inferência fossem consultadas pelo programa. O conhecimento que foi desejado recuperar da ontologia, é a de quais alimentos existentes em seu universo são alérgenos. Tendo isso, habilita o sistema a aplicar técnicas de PLN para detectar os alérgenos nas listas de ingredientes.

A detecção de alérgenos ocorre por meio de uma comparação de *strings*, utilizando três técnicas distintas de cálculo de similaridade para aumentar a precisão na identificação devido aos erros de texto provindos de digitação e OCR. Seria possível realizar o tratamento desses erros na etapa de preparação de dados, apenas necessitaria utilizar o *corpus* de um modelo como do FastText para checar palavras semelhantes em um dicionário, porém essa abordagem foge da proposta metodológica do trabalho.

Os cálculos de similaridade resultam em um valor limiar que varia de 0 a 100. As técnicas empregadas são: Distância de Levenshtein, Índice de Jaccard e Similaridade de Cosseno com *embeddings* gerados pelos modelos spaCy, FastText e BERT. Cada uma dessas técnicas é implementada utilizando bibliotecas para simplificar a codificação, otimizar o desempenho e precisão dos cálculos.

Antes do início da detecção de alérgenos no conjunto de testes, o sistema executa uma fase de otimização de hiperparâmetros. Durante esta fase, os três algoritmos são testados com todos os possíveis limiares de similaridade. O objetivo é identificar qual limiar proporciona o maior valor da métrica F1-Score para cada algoritmo. Considerando a relevância de um alto recall em um modelo voltado ao domínio das alergias alimentares, para assegurar a detecção abrangente de alérgenos, esta métrica inicialmente se apresenta como uma candidata de grande potencial a orientar a otimização do modelo. No entanto, um foco exclusivo no recall pode levar a um desbalanço, fazendo com que o modelo tenda a adotar um limiar de similaridade 0 como parâmetro. Essa sensibilidade excessiva aumentaria a incidência de falsos positivos, uma consequência indesejável na identificação de alérgenos. Para mitigar esse risco, tornou-se necessário avaliar também a precisão. Assim, a escolha recaiu sobre a otimização com base no F1-Score, uma métrica que harmoniza de maneira eficiente o equilíbrio entre recall e precisão, atendendo às necessidades específicas do modelo.

7.5. Avaliação

A avaliação do desempenho do sistema desenvolvido utiliza métricas que possibilitam uma análise objetiva quanto sua eficácia na detecção de alérgenos.

A Figura 4 e a Tabela 3 demonstram o comparativo dos resultados para cada métrica dos algoritmos trabalhados.

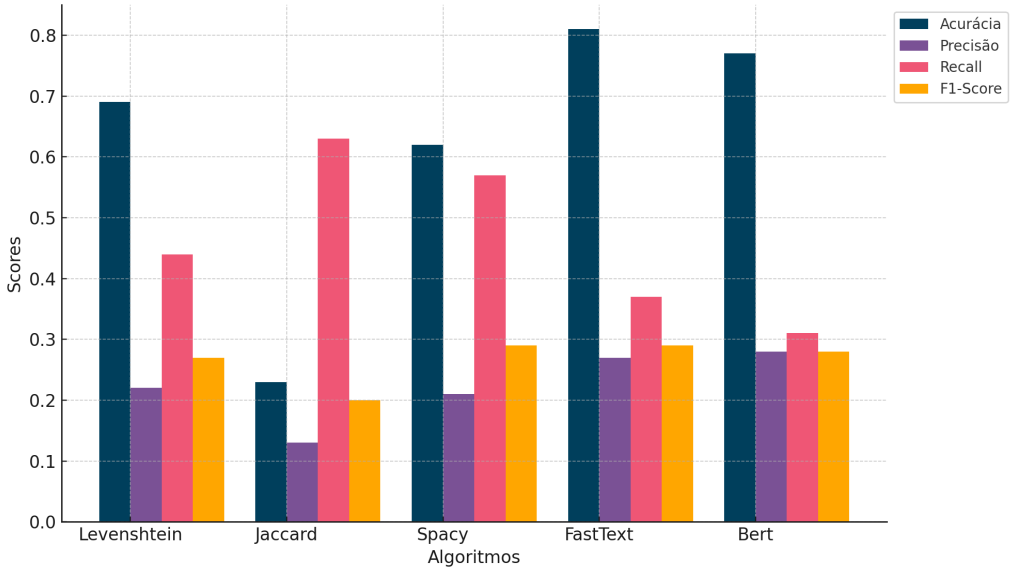

Figura 4. Comparativo dos resultados de cada algoritmo
Fonte: Autor do trabalho

Tabela 3. Comparativo dos resultados de cada algoritmo

Métrica/Algoritmo	Levenshtein	Jaccard	Spacy	FastText	Bert
Acurácia	0.69	0.23	0.62	0.81	0.77
Precisão	0.22	0.13	0.21	0.27	0.28
Recall	0.44	0.63	0.57	0.37	0.31
F1-Score	0.27	0.20	0.29	0.29	0.28

Com base na avaliação, o algoritmo FastText emerge como o mais promissor, exibindo a maior acurácia e um F1-Score equilibrado. BERT também mostra um desempenho sólido, com a maior precisão e um F1-Score competitivo. Jaccard, apesar de seu alto recall, tem desempenho inferior em outras métricas, sugerindo que pode não ser o mais adequado para esta tarefa. No entanto, todos os modelos apresentam desafios significativos em termos de precisão, indicando a necessidade de melhorias para reduzir os falsos negativos e melhorar a confiabilidade das predições.

8. Conclusão

A realização do projeto mostrou a viabilidade de um modelo para a detecção de alérgenos em produtos alimentícios, utilizando ontologias e aplicando técnicas de PLN diretamente na lista de ingredientes. Sendo assim, o objetivo geral foi atingido, marcando um avanço na direção de uma detecção automatizada e precisa de substâncias alergênicas.

Uma análise do estado da arte foi conduzida, proporcionando uma base conceitual para a implementação prática das técnicas de PLN escolhidas. Essas técnicas foram aplicadas em um conjunto de dados, permitindo a extração de características semânticas

relevantes para a tarefa. Além disso, foi desenvolvida uma representação ontológica, capturando o conhecimento essencial sobre as relações entre diferentes alimentos. Esse conhecimento, quando integrado ao modelo proposto, potencializou a identificação racional de alérgenos. Essa integração marcou uma contribuição para o campo, possibilitando o enriquecimento contínuo de uma ontologia de alérgenos através da identificação e incorporação de novos alérgenos e seus derivados.

A avaliação e análise revelaram a eficácia do modelo em conjunto com as diversas técnicas na detecção de alérgenos, validando assim a abordagem adotada. Uma comparação sistemática foi realizada entre as técnicas, gerando percepções valiosas sobre seus respectivos desempenhos e limitações, oferecendo uma compreensão mais profunda das oportunidades e desafios na detecção de alérgenos computacionalmente.

Em reflexão aos resultados obtidos, observa-se que o modelo FastText se sobressaiu conforme o método proposto. No entanto, é importante reconhecer as limitações do método. A escassez de dados na língua portuguesa impõe um desafio, visto que muitos dos dados disponíveis estão permeados por ruídos, exigindo uma fase de preparação de dados mais intensiva para mitigar impactos negativos tanto na evolução da ontologia quanto na precisão da detecção de alérgenos. Além disso, o modelo demonstra maior eficácia quando os ingredientes mencionam explicitamente os alérgenos inseridos inicialmente na ontologia. Quando os ingredientes são descritos de forma indireta ou com terminologias atípicas, o modelo enfrenta obstáculos significativos. Para aprimorar a eficácia do modelo, uma abordagem prática ainda a ser explorada é a classificação por comitê. Esse método consiste na combinação de múltiplos dos algoritmos e técnicas utilizadas, permitindo a sinergia de suas individualidades e culminando em uma classificação mais refinada e rigorosa. Diante desses aspectos, vislumbra-se um caminho para trabalhos futuros. A publicação dos resultados em plataformas de Web of Data pode ampliar a visibilidade e a aplicabilidade do modelo, assim como a integração com outras ontologias poderia enriquecer ainda mais o conhecimento representado. Ademais, há a possibilidade de adaptar e incorporar o modelo em aplicativos de software que, por meio de OCR, podem detectar alérgenos em rótulos de produtos alimentícios em tempo real, oferecendo uma ferramenta valiosa para consumidores com restrições alimentares.

Referências

DEVLIN, Jacob, et al. **BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding**. 2018.

LUGER, George. **Inteligência Artificial**. Tradução Daniel Vieira. 6 ed. São Paulo: Pearson, 2013.

MARTINEZ-PLUMED, Fernando; CONTRERAS-OCHANDO, Lidia; FERRI, Cesar; HERNANDEZ-ORALLO, Jose; KULL, Meelis; LACHICHE, Nicolas; RAMIREZ-QUINTANA, Maria Jose; FLACH, Peter. **CRISP-DM Twenty Years Later: from data mining processes to data science trajectories.** Ieee Transactions On Knowledge And Data Engineering, [S.L.], v. 33, n. 8, p. 3048-3061, 1 ago. 2021. Institute of Electrical and Electronics Engineers (IEEE).

MARTINS, Túlio Brandão Soares; REIS, Julio Cesar dos. **Addressing Inconsistencies in the DBPedia Evolution**. Campinas: Universidade Estadual de Campinas, 2019.

MINISTÉRIO DA SAÚDE. Constituição (2015). **Resolução da Diretoria Colegiada nº 26**, de 02 de julho de 2015. Brasília.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. Tradução Regina Célia Simille. 3 ed. Rio de Janeiro: Elsevier, 2013.

SHEARER, Rob; MOTIK, Boris; HORROCKS, Ian. **HermiT: A Highly-Efficient OWL Reasoner.** 2008.

YOUNG, Julio Christian; RUSLI, Andre. "**Review and Visualization of Facebook's FastText Pretrained Word Vector Model.**" 2019 International Conference on Engineering, Science, and Industrial Applications (ICESI), IEEE, 2019, pp. 1–6.

APÊNDICE B – CÓDIGO IMPLEMENTADO

algorithms.py

```python
from enum import Enum


class Algorithms(Enum):
    MATCH = "match"
    LEVENSHTEIN = "levenshtein"
    JACCARD = "jaccard"
    SPACY = "spacy"
    FASTTEXT = "fasttext"
    BERT = "bert"
```

cache.py

```python
import cachetools

cache = cachetools.LFUCache(maxsize=1000)


def get_cached_transformation(sentence, transformation_func):
    hash_key = hash(sentence)

    if hash_key in cache:
        return cache[hash_key]

    transformed = transformation_func(sentence)

    cache[hash_key] = transformed

    return transformed
```

data_preparation.py

```python
from tqdm import tqdm
```

```
from gerador_conjuntos import gerador_conjuntos


def preparar_dados():
    tqdm.pandas()
    gerador_conjuntos()
```

deteccao_alergenos.py

```
from algorithms import Algorithms
from similarity_metrics import levenshtein_distance,
↪  jaccard_similarity, spacy_similarity, \
    fasttext_similarity, bert_similarity


def is_allergen_present(ingredient, allergen, algorithm, threshold):
    match algorithm:
        case Algorithms.MATCH:
            return allergen in ingredient
        case Algorithms.LEVENSHTEIN:
            return levenshtein_distance(ingredient, allergen) >
            ↪  threshold
        case Algorithms.JACCARD:
            return jaccard_similarity(ingredient, allergen) >
            ↪  threshold
        case Algorithms.SPACY:
            return spacy_similarity(ingredient, allergen) > threshold
        case Algorithms.FASTTEXT:
            return fasttext_similarity(ingredient, allergen) >
            ↪  threshold
        case Algorithms.BERT:
            return bert_similarity(ingredient, allergen) > threshold


def detect_allergens(ingredients, cleaned_allergens_set,
↪  allergen_mapping, algorithm, threshold):
    detected_allergens = []
```

```
    for ingredient in ingredients:
        for cleaned_allergen in cleaned_allergens_set:
            if is_allergen_present(ingredient, cleaned_allergen,
            ↪  algorithm, threshold):
                original_allergen =
                ↪  allergen_mapping.get(cleaned_allergen,
                ↪  cleaned_allergen)
                detected_allergens.append((ingredient,
                ↪  original_allergen))
                break
    return detected_allergens
```

evaluation_metrics.py

```
from collections import namedtuple

import numpy as np

from extracao import extract_ingredients

evaluation_metrics = ['accuracy', 'precision', 'recall', 'f1']
PerformanceIndicators = namedtuple('PerformanceIndicators', ['TP',
↪  'FP', 'TN', 'FN', *evaluation_metrics])
EvaluationResult = namedtuple('EvaluationResult', ['confusion_mat',
↪  *evaluation_metrics])


def calculate_metrics(predicted, true_list, all_ingredients):
    predicted_set = set(predicted)
    true_set = set(true_list)
    all_ingredients_set = set(all_ingredients)

    TP = len(predicted_set & true_set)
    FP = len(predicted_set - true_set)
    FN = len(true_set - predicted_set)
    TN = len(all_ingredients_set - true_set - predicted_set)
```

```
    total = TP + FP + FN + TN
    accuracy = (TP + TN) / total if total != 0 else 0
    precision = TP / (TP + FP) if TP + FP != 0 else 0
    recall = TP / (TP + FN) if TP + FN != 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if precision
    ↪  + recall != 0 else 0

    return PerformanceIndicators(TP, FP, TN, FN, accuracy, precision,
    ↪  recall, f1)


def evaluate_algorithm(df, algorithm):
    indicators_list = []

    for row in df.itertuples():
        predicted = getattr(row, f'alergenos_{algorithm.value}')
        true_list = row.gabarito
        all_ingredients = extract_ingredients(row.ingredients_text_pt)

        indicators = calculate_metrics(predicted, true_list,
        ↪  all_ingredients)
        indicators_list.append(indicators)

    indicators_array = np.array(indicators_list)

    avg_indicators_values = np.mean(indicators_array, axis=0)
    avg_indicators = PerformanceIndicators(*avg_indicators_values)

    confusion_mat = np.array([[avg_indicators.TP, avg_indicators.FN],
    ↪  [avg_indicators.FP, avg_indicators.TN]])

    return EvaluationResult(confusion_mat, avg_indicators.accuracy,
    ↪  avg_indicators.precision, avg_indicators.recall,
                            avg_indicators.f1)
```

extracao.py

```
import re
```

```
def extract_ingredients(text):
    pattern = r"\s*,\s*|\s+e\s+(?=[a-zA-Z])"
    return [ingredient.strip() for ingredient in re.split(pattern,
    ↪  text)]
```

gerador_conjuntos.py

```
import pandas as pd

GRUPOS = {
    "facil": [
        {"code": "7894904219650", "gabarito": ['proteína de soja',
        ↪  'trigo']},
        {"code": "7896005804629", "gabarito": ['Leite em pó integral',
        ↪  'Soro de leite em pó']},
        {"code": "7896050411230",
         "gabarito": ['leite em pó integral', 'farinha de trigo
         ↪  enriquecida com ferro e ácido fólico',
                      'soro de leite em pó']},
        {"code": "7896068000075", "gabarito": ['leite', 'creme de
        ↪  leite', 'fermento lácteo']},
        {"code": "7897846901232", "gabarito": ['fibra de trigo',
        ↪  'farinha de soja', 'flocos de soja']},
        {"code": "7898067340404",
         "gabarito": ['Farinha de trigo enriquecida com ferro e ácido
         ↪  fólico', 'margarina', 'queijo mussarela',
                      'requeijão cremoso']},
        {"code": "7898409951824", "gabarito": ['soro de leite em pó',
        ↪  'leite em pó integral', 'ovo em pó']},
        {"code": "7898666240068", "gabarito": ['Leite de cabra']},
        {"code": "7898686950305", "gabarito": ['Proteína de soja']},
        {"code": "7897517206116", "gabarito": ['proteína texturizada
        ↪  de soja']}
    ],
```

```
    "medio": [
        {"code": "7891095006250", "gabarito": ['leite', 'soro de
        ↪  leite']},
        {"code": "7896022086114",
         "gabarito": ['Sêmola de trigo enriquecida com ferro e ácido
         ↪  fólico', 'sêmola de trigo durum',
                     'farelo de trigo', 'fibra de trigo']},
        {"code": "7896283005664", "gabarito": ['extrato de soja']},
        {"code": "7896423480894", "gabarito": []},
        {"code": "7898614670343", "gabarito": ['Amendoim triturado',
        ↪  'castanha triturada']},
        {"code": "7891103212420", "gabarito": ['FARINHA DE TRIGO
        ↪  ENRIQUECIDA COM FERRO E ÁCIDO FÓLICO']},
        {"code": "7894904261154", "gabarito": ['soja']},
        {"code": "7896024800602",
         "gabarito": ['farinha de trigo enriquecida com ferro e ácido
         ↪  fólico', 'farinha integral de soja',
                     'óleo essencial noz moscada']},
        {"code": "7897517206727", "gabarito": ['Extrato de soja', 'noz
        ↪  moscada']},
        {"code": "7898958161583", "gabarito": ['castanha de caju']}
    ],

    "dificil": [
        {"code": "5601038002209", "gabarito": ['Proteína de soja']},
        {"code": "7622300800529",
         "gabarito": ['farinha de trigo enriquecida com ferro e ácido
         ↪  fólico', 'farinha de trigo integral',
                     'aveia em flocos', 'farinha de cevada', 'farinha
                     ↪  de centeio', 'leite em pó desnatado']},
        {"code": "7891025115199", "gabarito": ['pasta de amêndoas']},
        {"code": "7891097000133",
         "gabarito": ['Leite desnatado e/ou leite desnatado
         ↪  reconstituído', 'creme de leite', 'leite em pó
         ↪  desnatado']},
        {"code": "7892840814175", "gabarito": ['soro de leite']},
        {"code": "7895000483730",
```

```
        "gabarito": ['Farinha de trigo enriquecida com ferro e ácido
        ↪ fólico', 'fibra de trigo', 'ovo em pó',
                    'clara de ovo em pó', 'Queijo em pó', 'composto
                    ↪ lácteo', 'leite em pó', 'soro de leite']},
        {"code": "7896003706598", "gabarito": ['FARINHA DE TRIGO
        ↪ FORTIFICADA COM FERRO E ÁCIDO FÓLICO', 'SOJA']},
        {"code": "7896080870274",
        "gabarito": ['FARINHA DE TRIGO ENRIQUECIDA COM FERRO E ÁCIDO
        ↪ FÓLICO', 'GORDURA VEGETAL HIDROGENADA DE SOJA']},
        {"code": "7896327514114", "gabarito": []},
        {"code": "7898215157311", "gabarito": ['Leite integral']}
    ]
}


def gerador_conjuntos():
    df = pd.read_csv(
        "openfoodfacts_export.csv",
        sep="\t",
        low_memory=False,
        usecols=["code", "product_name_pt", "ingredients_text_pt"]
    )

    all_codes = [item['code'] for group in GRUPOS.values() for item in
    ↪ group]
    present_codes = df['code'].unique().tolist()
    testing_codes = [code for code in all_codes if code in
    ↪ present_codes]

    if testing_codes:
        df_treinamento = df[~df['code'].isin(testing_codes)]

        gabaritos = [
            {'code': item['code'], 'gabarito': str(item['gabarito']),
            ↪ 'Grupo': grupo}
            for grupo, items in GRUPOS.items()
            for item in items
        ]
```

```
        df_teste = df.merge(pd.DataFrame(gabaritos), on='code',
        ↪  how='inner')

        df_teste['Grupo'] = pd.Categorical(df_teste['Grupo'],
        ↪  ["facil", "medio", "dificil"])
        df_teste = df_teste.sort_values('Grupo')

        df_treinamento.to_csv("conjunto_treinamento.csv", sep='\t',
        ↪  index=False)
        df_teste.to_csv("conjunto_teste.csv", sep='\t', index=False)
```

main.py

```
from IPython.core.display_functions import display

from data_preparation import preparar_dados
from teste import teste, load_and_preprocess_sample
from treinamento import treinamento, load_and_preprocess_data


def main():
    preparar_dados()
    df_treinamento = load_and_preprocess_data()
    cleaned_allergens_set, allergen_mapping =
    ↪  treinamento(df_treinamento)
    df_teste, extracted_ingredients_amostra =
    ↪  load_and_preprocess_sample()
    teste(extracted_ingredients_amostra, cleaned_allergens_set,
    ↪  allergen_mapping, df_teste)

    return df_teste


if __name__ == '__main__':
    result_df = main()
    display(result_df)
```

ontologia.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:food="http://www.ufsc.br/food#">
    <Ontology/>


    <!--

    //
    // Object Properties
    //

     -->


    <!-- http://www.ufsc.br/food#eDerivadoDe -->

    <ObjectProperty rdf:about="http://www.ufsc.br/food#eDerivadoDe">
        <rdfs:domain rdf:resource="http://www.ufsc.br/food#Alimento"/>
        <rdfs:range rdf:resource="http://www.ufsc.br/food#Alimento"/>
        <rdfs:label>é derivado de</rdfs:label>
    </ObjectProperty>


    <!--

    //
    // Classes
```

```
    //

     -->


    <!-- http://www.ufsc.br/food#Alergeno -->

    <Class rdf:about="http://www.ufsc.br/food#Alergeno">
        <rdfs:subClassOf
        ↪  rdf:resource="http://www.ufsc.br/food#Alimento"/>
        <rdfs:label>Alérgeno</rdfs:label>
    </Class>



    <!-- http://www.ufsc.br/food#AlergenoPorDerivacao -->

    <Class rdf:about="http://www.ufsc.br/food#AlergenoPorDerivacao">
        <equivalentClass>
            <Restriction>
                <onProperty
                ↪  rdf:resource="http://www.ufsc.br/food#eDerivadoDe"/>
                <someValuesFrom
                ↪  rdf:resource="http://www.ufsc.br/food#Alergeno"/>
            </Restriction>
        </equivalentClass>
        <rdfs:subClassOf
        ↪  rdf:resource="http://www.ufsc.br/food#Alimento"/>
        <rdfs:label>Alérgeno por Derivação</rdfs:label>
    </Class>



    <!-- http://www.ufsc.br/food#Alimento -->

    <Class rdf:about="http://www.ufsc.br/food#Alimento">
        <rdfs:label>Alimento</rdfs:label>
    </Class>
```

```
<!-- http://www.ufsc.br/food#Crustáceo -->

<Class rdf:about="http://www.ufsc.br/food#Crustáceo">
    <rdfs:subClassOf
    ↪  rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Crustáceo</rdfs:label>
</Class>


<!--

//
// Individuals
//

 -->


<!-- http://www.ufsc.br/food#Amendoa -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Amendoa">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Amêndoa</rdfs:label>
</NamedIndividual>


<!-- http://www.ufsc.br/food#Amendoim -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Amendoim">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Amendoim</rdfs:label>
</NamedIndividual>
```

```
<!-- http://www.ufsc.br/food#Aveia -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Aveia">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Aveia</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Avelã -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Avelã">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Avelã</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Camarão -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Camarão">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Crustáceo"/>
    <rdfs:label>Camarão</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Caranguejo -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Caranguejo">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Crustáceo"/>
    <rdfs:label>Caranguejo</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Castanha -->
```

```
<NamedIndividual rdf:about="http://www.ufsc.br/food#Castanha">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Castanha</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Castanha-de-caju -->

<NamedIndividual
↪  rdf:about="http://www.ufsc.br/food#Castanha-de-caju">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Castanha-de-caju</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Castanha-do-brasil -->

<NamedIndividual
↪  rdf:about="http://www.ufsc.br/food#Castanha-do-brasil">
    <sameAs
    ↪  rdf:resource="http://www.ufsc.br/food#Castanha-do-pará"/>
    <rdfs:label>Castanha-do-brasil</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Castanha-do-pará -->

<rdf:Description
↪  rdf:about="http://www.ufsc.br/food#Castanha-do-pará"/>



<!-- http://www.ufsc.br/food#Castanha-do-pará -->
```

```
<NamedIndividual
↪  rdf:about="http://www.ufsc.br/food#Castanha-do-pará">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Castanha-do-pará</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Centeio -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Centeio">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Centeio</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Cevada -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Cevada">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Cevada</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Lagosta -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Lagosta">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Crustáceo"/>
    <rdfs:label>Lagosta</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Lagostim -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Lagostim">
```

```
        <rdf:type rdf:resource="http://www.ufsc.br/food#Crustáceo"/>
        <rdfs:label>Lagostim</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Leite -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Leite">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
        <rdfs:label>Leite</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Látex_natural -->

    <NamedIndividual
    ↪   rdf:about="http://www.ufsc.br/food#Látex_natural">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
        <rdfs:label>Látex natural</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Macadamia -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Macadamia">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
        <rdfs:label>Macadâmia</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Nozes -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Nozes">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
```

```
    <rdfs:label>Nozes</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Ovo -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Ovo">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Ovo</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Pecã -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Pecã">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Pecã</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Peixe -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Peixe">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Peixe</rdfs:label>
</NamedIndividual>



<!-- http://www.ufsc.br/food#Pistache -->

<NamedIndividual rdf:about="http://www.ufsc.br/food#Pistache">
    <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
    <rdfs:label>Pistache</rdfs:label>
</NamedIndividual>
```

```
    <!-- http://www.ufsc.br/food#Siri -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Siri">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Crustáceo"/>
        <rdfs:label>Siri</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Soja -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Soja">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
        <rdfs:label>Soja</rdfs:label>
    </NamedIndividual>



    <!-- http://www.ufsc.br/food#Trigo -->

    <NamedIndividual rdf:about="http://www.ufsc.br/food#Trigo">
        <rdf:type rdf:resource="http://www.ufsc.br/food#Alergeno"/>
        <rdfs:label>Trigo</rdfs:label>
    </NamedIndividual>
</rdf:RDF>
```

ontology_operations.py

```
from owlready2 import default_world, get_ontology, sync_reasoner

ONTOLOGY_PATH = "ontologia.owl"
LOAD_ALLERGENS_QUERY_PATH = "query_alergenos.sparql"

ontology = get_ontology(ONTOLOGY_PATH).load()
```

```
sync_reasoner()


def load_allergens_from_ontology():
    with open(LOAD_ALLERGENS_QUERY_PATH, "r") as file:
        query = file.read()
    results = list(default_world.sparql(query))
    allergens = set(label for [label] in results)
    return allergens


def create_allergen_in_ontology(derived_name, allergen_base):
    allergen_base_instance = ontology.search_one(label=allergen_base)
    Alimento = ontology.search_one(label="Alimento")

    new_allergen = Alimento(derived_name.title().replace(" ", ""))
    new_allergen.label = [derived_name.title()]
    new_allergen.eDerivadoDe.append(allergen_base_instance)
```

data_preparation.py

```
from tqdm import tqdm

from gerador_conjuntos import gerador_conjuntos


def preparar_dados():
    tqdm.pandas()
    gerador_conjuntos()
```

plot.py

```
from matplotlib import pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay


def plot_confusion_matrix(matrix, algorithm):
```

```
    class_labels = ['Alérgeno', 'Não alergeno']
    disp = ConfusionMatrixDisplay(confusion_matrix=matrix,
    ↪  display_labels=class_labels)
    disp.plot(cmap='Purples', values_format='.2f')

    ax = plt.gca()
    ax.set_yticklabels(class_labels, rotation=90, va='center')

    plt.xlabel('Rótulo predito')
    plt.ylabel('Rótulo verdadeiro')
    plt.title(f'Matriz de confusão para o algoritmo:
    ↪  {algorithm.value}')

    plt.tight_layout()
    plt.show()


def plot_metrics(avg_accuracy, avg_precisions, avg_recalls, avg_f1,
↪  algorithm):
    metrics = ['Acurácia', 'Precisão', 'Recall', 'F1-Score']
    values = [avg_accuracy, avg_precisions, avg_recalls, avg_f1]

    plt.figure(figsize=(10, 6))
    bars = plt.bar(metrics, values, color=['#003f5c', '#7a5195',
    ↪  '#ef5675', '#ffa600'])

    plt.ylabel('Pontuação')
    plt.title(f'Métricas para o algoritmo: {algorithm}')
    plt.ylim(0, 1)

    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2.0, yval,
        ↪  round(yval, 2), ha='center', va='bottom')

    plt.show()
```

pre_processamento.py

```python
import re

from unidecode import unidecode


def clean_text(text):
    cleaned = unidecode(text.lower().strip())
    cleaned = re.sub(r'[^a-zA-Z\s,]', ' ', cleaned)
    cleaned = re.sub(r',+', ',', cleaned)
    cleaned = re.sub(r' +', ' ', cleaned)
    return cleaned


def preprocess_data(df):
    df.dropna(subset=['ingredients_text_pt'], inplace=True)
    df['ingredients_text_pt'] =
    ↪  df['ingredients_text_pt'].apply(clean_text)
    return df
```

query_alergenos.sparql

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX food: <http://www.ufsc.br/food#>

SELECT ?label
WHERE {
    ?individuo rdfs:label ?label .
    {
        ?individuo rdf:type food:Alergeno .
    }
    UNION
    {
        ?individuo rdf:type food:AlergenoPorDerivacao .
        ?individuo rdfs:label ?label .
    }
```

```
    UNION
    {
        ?individuo rdf:type ?subClass .
        ?subClass rdfs:subClassOf+ food:Alergeno .
    }
}
```

similarity_metrics.py

```
import numpy as np
import spacy
import fasttext.util
from thefuzz import fuzz
from transformers import BertTokenizer, BertModel

from torch.nn.functional import cosine_similarity as
↪  cosine_similarity_torch
from sklearn.metrics.pairwise import cosine_similarity as
↪  cosine_similarity_sklearn

from cache import get_cached_transformation

model_spacy = spacy.load('pt_core_news_md')

MODEL_LANG = "pt"
fasttext.util.download_model(MODEL_LANG, if_exists='ignore')
model_fasttext = fasttext.load_model(f'cc.{MODEL_LANG}.300.bin')

BERT_ARCH = "neuralmind/bert-large-portuguese-cased"
tokenizer_bert = BertTokenizer.from_pretrained(BERT_ARCH)
model_bert = BertModel.from_pretrained(BERT_ARCH)


def normalize_thresold(value):
    return value * 100
```

```
def levenshtein_distance(ingredient, allergen):
    return fuzz.ratio(ingredient, allergen)


def jaccard_similarity(ingredient, allergen):
    ingredient_set = set(ingredient)
    allergen_set = set(allergen)
    intersection = ingredient_set.intersection(allergen_set)
    union = ingredient_set.union(allergen_set)
    return normalize_thresold(len(intersection) / len(union))


def spacy_similarity(ingredient, allergen):
    token1 = get_cached_transformation(ingredient, model_spacy)
    token2 = get_cached_transformation(allergen, model_spacy)

    if token1.has_vector and token2.has_vector:
        similarity = normalize_thresold(token1.similarity(token2))
    else:
        similarity = 0

    return similarity


def sentence_to_vector_fasttext(sentence):
    words = sentence.split()
    vectors = [model_fasttext.get_word_vector(word) for word in words]

    if vectors:
        return np.mean(vectors, axis=0)
    else:
        vector_dim = model_fasttext.get_dimension()
        return np.zeros(vector_dim)


def fasttext_similarity(ingredient, allergen):
    token1 = get_cached_transformation(ingredient,
    ↪  sentence_to_vector_fasttext)
```

```
    token2 = get_cached_transformation(allergen,
    ↪  sentence_to_vector_fasttext)
    token1 = np.array(token1).reshape(1, -1)
    token2 = np.array(token2).reshape(1, -1)
    return normalize_thresold(cosine_similarity_sklearn(token1,
    ↪  token2))


def sentences_to_embeddings_bert(sentences):
    inputs = tokenizer_bert(sentences, return_tensors="pt",
                            padding=True, truncation=True,
                            ↪  max_length=128)
    outputs = model_bert(**inputs)
    return outputs.last_hidden_state.mean(dim=1)


def bert_similarity(ingredient, allergen):
    ingredient_embedding = get_cached_transformation(ingredient,
    ↪  sentences_to_embeddings_bert)
    allergen_embedding = get_cached_transformation(allergen,
    ↪  sentences_to_embeddings_bert)

    return
    ↪  normalize_thresold(cosine_similarity_torch(ingredient_embedding,
    ↪  allergen_embedding))
```

teste.py

```
import ast

import pandas as pd

from algorithms import Algorithms
from cache import cache
from deteccao_alergenos import detect_allergens
from evalutation_metrics import evaluate_algorithm
from extracao import extract_ingredients
from plot import plot_confusion_matrix, plot_metrics
```

```
from pre_processamento import preprocess_data, clean_text
from similarity_metrics import model_fasttext


def find_best_threshold(algorithm, extracted_ingredients_amostra,
↪  cleaned_allergens_set, allergen_mapping, df_teste):
    best_threshold = 0
    best_f1 = 0
    for threshold in range(0, 101):
        detected_allergens =
        ↪  extracted_ingredients_amostra.progress_apply(
            lambda x: detect_allergens(x, cleaned_allergens_set,
            ↪  allergen_mapping, algorithm, threshold)
        )
        df_teste[f'alergenos_{algorithm.value}'] =
        ↪  detected_allergens.apply(lambda x: [detected[0] for
        ↪  detected in x])
        metrics = evaluate_algorithm(df_teste, algorithm)
        avg_f1 = metrics.f1
        if avg_f1 > best_f1:
            best_f1 = avg_f1
            best_threshold = threshold
    return best_threshold


def apply_best_threshold(algorithm, best_threshold,
↪  extracted_ingredients_amostra, cleaned_allergens_set,
                        allergen_mapping, df_teste):
    detected_allergens = extracted_ingredients_amostra.progress_apply(
        lambda x: detect_allergens(x, cleaned_allergens_set,
        ↪  allergen_mapping, algorithm, best_threshold)
    )
    df_teste[f'alergenos_{algorithm.value}'] =
    ↪  detected_allergens.apply(lambda x: [detected[0] for detected
    ↪  in x])
```

```
def plotar_graficos(best_confusion_mat, algorithm, best_avg_accuracy,
↪  best_avg_precisions, best_avg_recalls,
                    best_avg_f1):
    plot_confusion_matrix(best_confusion_mat, algorithm)
    plot_metrics(best_avg_accuracy, best_avg_precisions,
    ↪  best_avg_recalls, best_avg_f1, algorithm.value)


def evaluate_and_visualize(algorithm, df_teste):
    best_confusion_mat, best_avg_accuracy, best_avg_precisions,
    ↪  best_avg_recalls, best_avg_f1 = evaluate_algorithm(
        df_teste, algorithm)
    plotar_graficos(best_confusion_mat, algorithm, best_avg_accuracy,
    ↪  best_avg_precisions, best_avg_recalls,
                    best_avg_f1)


def load_and_preprocess_sample():
    df_teste = pd.read_csv('conjunto_teste.csv', sep="\t",
    ↪  low_memory=False,
                           usecols=["code", "product_name_pt",
                           ↪  "ingredients_text_pt", "gabarito"])

    df_teste = preprocess_data(df_teste)

    extracted_ingredients_amostra =
    ↪  df_teste['ingredients_text_pt'].progress_apply(extract_ingredients)

    df_teste['gabarito'] =
    ↪  df_teste['gabarito'].apply(ast.literal_eval).apply(lambda x:
    ↪  [clean_text(i) for i in x])
    return df_teste, extracted_ingredients_amostra


def teste(extracted_ingredients_amostra, cleaned_allergens_set,
↪  allergen_mapping, df_teste):
    for algorithm in Algorithms:
```

```
        if algorithm == Algorithms.FASTTEXT and model_fasttext is
        ↪  None:
            continue

        best_threshold = find_best_threshold(algorithm,
        ↪  extracted_ingredients_amostra, cleaned_allergens_set,
                                             allergen_mapping,
                                             ↪  df_teste)
        apply_best_threshold(algorithm, best_threshold,
        ↪  extracted_ingredients_amostra, cleaned_allergens_set,
                               allergen_mapping, df_teste)
        evaluate_and_visualize(algorithm, df_teste)

        cache.clear()
```

treinamento.py

```
import pandas as pd

from algorithms import Algorithms
from deteccao_alergenos import detect_allergens
from extracao import extract_ingredients
from ontology_operations import load_allergens_from_ontology,
↪  create_allergen_in_ontology, ontology, ONTOLOGY_PATH
from pre_processamento import clean_text, preprocess_data


def consultar_alergenos():
    allergens_set = load_allergens_from_ontology()
    cleaned_allergens_set = set()
    allergen_mapping = {}

    for allergen in allergens_set:
        cleaned = clean_text(allergen)
        cleaned_allergens_set.add(cleaned)
        allergen_mapping[cleaned] = allergen
    return cleaned_allergens_set, allergen_mapping
```

```
def detectar_alergenos_conjunto_treinamento(df, cleaned_allergens_set,
↪  allergen_mapping):
    extracted_ingredients =
    ↪  df['ingredients_text_pt'].progress_apply(extract_ingredients)
    detected_allergens_match = extracted_ingredients.progress_apply(
        lambda x: detect_allergens(x, cleaned_allergens_set,
        ↪  allergen_mapping, Algorithms.MATCH, 0)
    )
    return detected_allergens_match


def popular_ontologia(detected_allergens_match):
    added_derived_allergens = set()

    for allergen_pairs in detected_allergens_match:
        for derivated_allergen, base_allergen in allergen_pairs:
            if derivated_allergen != base_allergen and
            ↪  derivated_allergen not in added_derived_allergens and
            ↪  not ontology.search_one(
                    label=derivated_allergen.title()):
                create_allergen_in_ontology(derivated_allergen,
                ↪  base_allergen)
                added_derived_allergens.add(derivated_allergen)
    ontology.save(ONTOLOGY_PATH)


def load_and_preprocess_data():
    df = pd.read_csv('conjunto_treinamento.csv', sep="\t",
    ↪  low_memory=False,
                     usecols=["product_name_pt",
                     ↪  "ingredients_text_pt"])
    df = preprocess_data(df)
    return df


def treinamento(df):
```

```
    cleaned_allergens_set, allergen_mapping = consultar_alergenos()

    detected_allergens_match =
    ↪  detectar_alergenos_conjunto_treinamento(df,
    ↪  cleaned_allergens_set, allergen_mapping)

    popular_ontologia(detected_allergens_match)

    return cleaned_allergens_set, allergen_mapping
```